AP-701 Fundamentals of Aircraft Design
Main Guide

Cap Eng **Ney** Rafael Sêcco
ney@ita.br

August 10, 2021

1 Introduction

We need to develop an analysis cycle that computes aircraft performance metrics based on design variables. This way, we can vary these design variables and verify their impact on the aircraft metrics. This even allows the use of optimization techniques to find the aircraft that best matches the design requirements. This guide introduces the main components of this analysis cycle and their interfaces.

2 Analysis cycle overview

The aircraft analysis cycle must be composed by modules organized in a logical manner. This structure should steadily increase information about this design. Due to the complexity of aircraft design analysis, internal iterative cycles are necessary to solve the problem.

For this course we will use the analysis cycle described in Fig. 1. This cycle takes design variables that should be chosen by the designer (such as geometric parameters of the aircraft, flap types, and number of engines) and returns performance metrics (such as Maximum Takeoff Weight (MTOW), required thrust, and static margin).

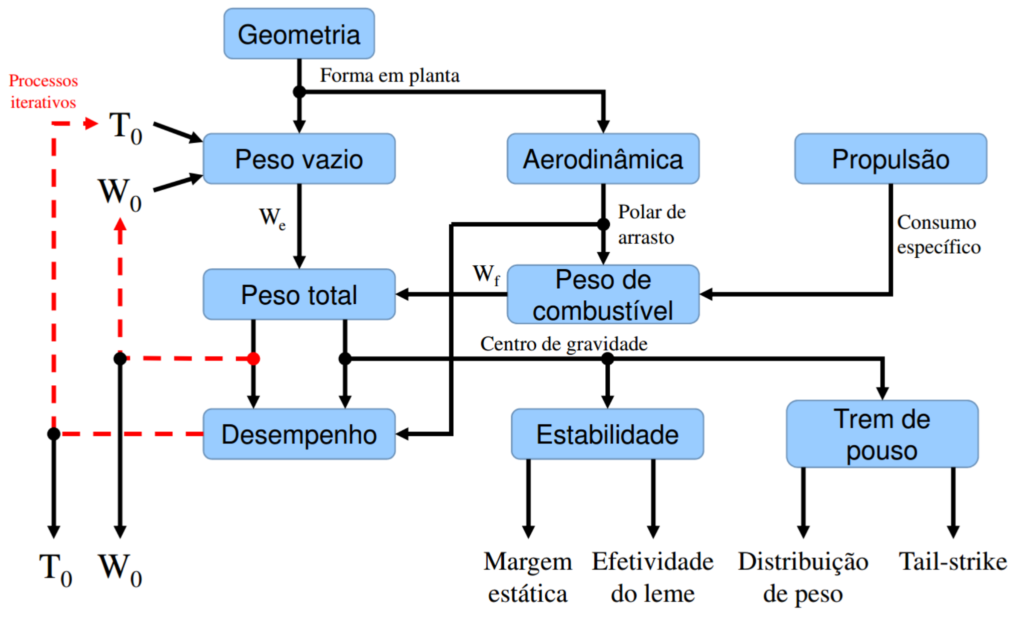


Figure 1: Aircraft analysis cycle used in this course.

There are two iterative process within this analysis cycle. The first one is necessary to determine the MTOW (or W_0) and the second one is used to compute the required takeoff thrust (T_0).

2.1 Programming structure

First create a folder to hold all scripts of this project. I will refer to this folder as *main folder* throughout this document.

Add to this folder the `designTool.py` script provided for this course. You will implement the new modules in the appropriate spaces of this file.

Create another Python script (for instance, `homework.py`) inside the main folder. You will implement the necessary function calls to solve you homework problems in this script. You can access the functions from the `designTools.py` file by import it at the beginning of the script. For example, if you want to run the `atmosphere` function in your new script, you can use:

```
import designTools as dt
T,p,rho,mi = dt.atmoshpere(10000)
print(T)
```

I suggest that you have a script for the homework assignment and another script to design your aircraft.

Note that several constants and conversion factors are defined at the beginning of the `designTool.py` file. Make sure you use these parameters to obtain the same results of the test cases presented here.

The initial version of the `designTool.py` file indicates where you should insert your code. For instance, let's look at the `geometry` function:

```
def geometry(airplane):

    # Unpack dictionary
    S_w = airplane['S_w']
    AR_w = airplane['AR_w']
    taper_w = airplane['taper_w']
    sweep_w = airplane['sweep_w']
    dihedral_w = airplane['dihedral_w']
    xr_w = airplane['xr_w']
    zr_w = airplane['zr_w']
    Cht = airplane['Cht']
    AR_h = airplane['AR_h']
    taper_h = airplane['taper_h']
    sweep_h = airplane['sweep_h']
    dihedral_h = airplane['dihedral_h']
    Lc_h = airplane['Lc_h']
    zr_h = airplane['zr_h']
    Cvt = airplane['Cvt']
    AR_v = airplane['AR_v']
    taper_v = airplane['taper_v']
    sweep_v = airplane['sweep_v']
    Lb_v = airplane['Lb_v']
    zr_v = airplane['zr_v']

    ### ADD CODE FROM SECTION 3.1 HERE ###


    # Update dictionary with new results
    airplane['b_w'] = b_w
    airplane['cr_w'] = cr_w
    airplane['xt_w'] = xt_w
    airplane['yt_w'] = yt_w
    airplane['zt_w'] = zt_w
    airplane['ct_w'] = ct_w
    airplane['xm_w'] = xm_w
    airplane['ym_w'] = ym_w
    airplane['zm_w'] = zm_w
    airplane['cm_w'] = cm_w
    airplane['S_h'] = S_h
    airplane['b_h'] = b_h
    airplane['xr_h'] = xr_h
    airplane['cr_h'] = cr_h
    airplane['xt_h'] = xt_h
    airplane['yt_h'] = yt_h
    airplane['zt_h'] = zt_h
    airplane['ct_h'] = ct_h
    airplane['xm_h'] = xm_h
    airplane['ym_h'] = ym_h
    airplane['zm_h'] = zm_h
    airplane['cm_h'] = cm_h
    airplane['S_v'] = S_v
    airplane['b_v'] = b_v
    airplane['xr_v'] = xr_v
    airplane['cr_v'] = cr_v
    airplane['xt_v'] = xt_v
    airplane['zt_v'] = zt_v
    airplane['ct_v'] = ct_v
    airplane['xm_v'] = xm_v
    airplane['zm_v'] = zm_v
    airplane['cm_v'] = cm_v

    # All variables are stored in the dictionary.
    # There is no need to return anything
    return None
```

The `designTool` package uses a dictionary named `airplane` to carry information among its several functions. The function retrieves the necessary variables at the beginning and stores the new information at the end. The initial version of the `designTool.py` code already have the necessary dictionary interfaces of all functions. You may add new dictionary keys if you want to store additional information, BUT DO NOT REMOVE ANY OF THE DEFINITIONS ALREADY IN PLACE OR THE CODE WILL BREAK.

The exercises of this course only require that you modify the inner portion of the functions, as indicated by the comments (look for the lines with: `### ADD CODE FROM SECTION ... HERE ###`).

3 Analysis modules

In this section we describe each module of Fig. 1.

3.1 Geometry module

The geometry module is responsible for taking dimensional and non-dimensional geometric parameters and convert them into a three-dimensional description of the aircraft. For example, we should take geometric parameters, such as wing aspect ratio (AR_w) and vertical tail volume coefficient (C_{VT}), and determine the x, y, and z coordinates of the root and tip sections of all aerodynamic surfaces.

At this point we assume that the designer already developed the cabin layout studies to determine the fuselage length (L_f) and diameter (D_f).

3.1.1 Inputs

The dictionary inputs of this module are:

- Wing geometric parameters: S_w (`S_w`), AR_w (`AR_w`), λ_w (`taper_w`), Λ_w (`sweep_w`), δ_w (`dihedral_w`), $x_{r,w}$ (`xr_w`), and $z_{r,w}$ (`zr_w`).
- Horizontal tail geometric parameters: C_{ht} (`Cht`), AR_h (`AR_h`), λ_h (`taper_h`), Λ_h (`sweep_h`), δ_h (`dihedral_h`), $L_h/c_{m,w}$ (`Lc_h`), and $z_{r,h}$ (`zr_h`).
- Vertical tail geometric parameters: C_{vt} (`Cvt`), AR_v (`AR_v`), λ_v (`taper_v`), Λ_v (`sweep_v`), L_v/b_w (`Lb_v`), and $z_{r,v}$ (`zr_v`).

3.1.2 Outputs

The dictionary outputs of this module are:

- Wing dimensional parameters: b_w (`b_w`), $c_{r,w}$ (`cr_w`), $x_{t,w}$ (`xt_w`), $y_{t,w}$ (`yt_w`), $z_{t,w}$ (`zt_w`), $c_{t,w}$ (`ct_w`), $x_{m,w}$ (`xm_w`), $y_{m,w}$ (`ym_w`), $z_{m,w}$ (`zm_w`), and $c_{m,w}$ (`cm_w`).

- Horizontal tail dimensional parameters: S_h (S_h), b_h (b_h), $x_{r,h}$ (xr_h), $c_{r,h}$ (cr_h), $x_{t,h}$ (xt_h), $y_{t,h}$ (yt_h), $z_{t,h}$ (zt_h), $c_{t,h}$ (ct_h), $x_{m,h}$ (xm_h), $y_{m,h}$ (ym_h), $z_{m,h}$ (zm_h), and $c_{m,h}$ (cm_h).
- Vertical tail dimensional parameters: S_v (S_v), b_v (b_v), $x_{r,v}$ (xr_v), $c_{r,v}$ (cr_v), $x_{t,v}$ (xt_v), $z_{t,v}$ (zt_v), $c_{t,v}$ (ct_v), $x_{m,v}$ (xm_v), $z_{m,v}$ (zm_v), and $c_{m,v}$ (cm_v).

3.1.3 Sizing of the wing

In this subsection we will explain how to size the wing based on its area (S_w) and nondimensional parameters (aspect ratio AR_w, taper ratio λ_w, sweep at the quarter-chord Λ_w, and dihedral δ_w).

First we can use the aspect ratio definition to compute the span b_w:

$$b_w = \sqrt{AR_w \cdot S_w} \tag{1}$$

Then we can compute the root chord $c_{r,w}$ with:

$$c_{r,w} = \frac{2S_w}{b_w(1+\lambda_w)} \tag{2}$$

The tip chord can be determined with:

$$c_{t,w} = \lambda_w \cdot c_{r,w} \tag{3}$$

Now we will compute the coordinates of the wing tip leading edge with respect to the fuselage nose. The wing span already defines the wing tip lateral position:

$$y_{t,w} = \frac{b_w}{2} \tag{4}$$

Since the sweep is taken at the quarter-chord, the longitudinal position of the wing tip is given by:

$$x_{t,w} = x_{r,w} + y_{t,w}\tan\Lambda_w + \frac{c_{r,w} - c_{t,w}}{4} \tag{5}$$

We can finally determine the vertical location of the wing tip leading edge with the dihedral:

$$z_{t,w} = z_{r,w} + y_{t,w}\tan\delta_w \tag{6}$$

Next we compute the mean aerodynamic chord of the wing with:

$$c_{m,w} = \frac{2c_{r,w}}{3} \cdot \frac{1+\lambda_w+\lambda_w^2}{1+\lambda_w} \tag{7}$$

The lateral position of the mean aerodynamic chord can be computed with:

$$y_{m,w} = \frac{b_w}{6} \cdot \frac{1+2\lambda_w}{1+\lambda_w} \tag{8}$$

The other coordinates of the mean aerodynamic chord leading edge can be computed by using sweep and dihedral relationships at this intermediate span location:

$$x_{m,w} = x_{r,w} + y_{m,w}\tan\Lambda_w + \frac{c_{r,w} - c_{m,w}}{4} \tag{9}$$

$$z_{m,w} = z_{r,w} + y_{m,w}\tan\delta_w \tag{10}$$

3.1.4 Sizing of the horizontal tail

First we need to determine the horizontal tail area based on its volume coefficient.

From the given nondimensional tail lever definition we can get:

$$L_h = \frac{L_h}{c_{m,w}} \cdot c_{m,w} \tag{11}$$

The tail volume coefficient for the horizontal tail gives:

$$S_h = \frac{S_w \cdot c_{m,w}}{L_h} \cdot C_{ht} \tag{12}$$

Now that we know the horizontal tail area, we can repeat most of the sizing process of the wing. First we can use the aspect ratio definition to compute the span b_h:

$$b_h = \sqrt{AR_h \cdot S_h} \tag{13}$$

Then we can compute the root chord $c_{r,h}$ with:

$$c_{r,h} = \frac{2S_h}{b_h(1+\lambda_h)} \tag{14}$$

The tip chord can be determined with:

$$c_{t,h} = \lambda_h \cdot c_{r,h} \tag{15}$$

This time we need to define the mean aerodynamic chord properties since the tail lever is defined with respect to this quantity. The mean aerodynamic chord of the horizontal tail is:

$$c_{m,h} = \frac{2c_{r,h}}{3} \cdot \frac{1+\lambda_h+\lambda_h^2}{1+\lambda_h} \tag{16}$$

Then the longitudinal position of the leading edge of the horizontal tail mean aerodynamic chord is:

$$x_{m,h} = x_{m,w} + L_h + \frac{c_{m,w} - c_{m,h}}{4} \tag{17}$$

The lateral position of the mean aerodynamic chord can be computed with:

$$y_{m,h} = \frac{b_h}{6} \cdot \frac{1+2\lambda_h}{1+\lambda_h} \tag{18}$$

The vertical position of the mean aerodynamic chord is given by:

$$z_{m,h} = z_{r,h} + y_{m,h} \tan \delta_h \tag{19}$$

Now that we know the coordinates of the mean aerodynamic chord leading edge, we can use the sweep relationship to find the root leading edge position:

$$x_{r,h} = x_{m,h} - y_{m,h} \tan \Lambda_h + \frac{c_{m,h} - c_{r,h}}{4} \tag{20}$$

We can finally determine the coordinates of the horizontal tail tip leading edge with the same expressions that we used for the wing:

$$y_{t,h} = \frac{b_h}{2} \tag{21}$$

$$x_{t,h} = x_{r,h} + y_{t,h} \tan \Lambda_h + \frac{c_{r,h} - c_{t,h}}{4} \tag{22}$$

$$z_{t,h} = z_{r,h} + y_{t,h} \tan \delta_h \tag{23}$$

3.1.5 Sizing of the vertical tail

Special attention is necessary for this section since the spanwise direction of the vertical tail is along the z direction instead of the y direction. In addition, the vertical tail is not symmetric with respect to its root section.

First we need to determine the vertical tail area based on its volume coefficient. From the given nondimensional tail lever definition we can get:

$$L_v = \frac{L_v}{b_w} \cdot b_w \tag{24}$$

The tail volume coefficient for the vertical tail gives:

$$S_v = \frac{S_w \cdot b_w}{L_v} \cdot C_{vt} \tag{25}$$

Now that we know the vertical tail area, we can repeat most of the sizing process of the wing. First we can use the aspect ratio definition to compute the span b_v:

$$b_v = \sqrt{AR_v \cdot S_v} \tag{26}$$

Then we can compute the root chord $c_{r,v}$ with:

$$c_{r,v} = \frac{2S_v}{b_v(1+\lambda_v)} \tag{27}$$

The tip chord can be determined with:

$$c_{t,v} = \lambda_v \cdot c_{r,v} \tag{28}$$

This time we need to define the mean aerodynamic chord properties since the tail lever is defined with respect to this quantity. The mean aerodynamic chord of the vertical tail is:

$$c_{m,v} = \frac{2c_{r,v}}{3} \cdot \frac{1+\lambda_v+\lambda_v^2}{1+\lambda_v} \tag{29}$$

Then the longitudinal position of the leading edge of the vertical tail mean aerodynamic chord is:

$$x_{m,v} = x_{m,w} + L_v + \frac{c_{m,w} - c_{m,v}}{4} \tag{30}$$

The vertical position of the mean aerodynamic chord can be computed with:

$$z_{m,v} = z_{r,v} + \frac{b_v}{3} \cdot \frac{1+2\lambda_v}{1+\lambda_v} \tag{31}$$

Note that we had to modify the expression to consider the span of the vertical tail as the half span of an equivalent surface with root symmetry. Now that we know the coordinates of the mean aerodynamic chord leading edge, we can use the sweep relationship to find the root leading edge position:

$$x_{r,v} = x_{m,v} - (z_{m,v} - z_{r,v}) \tan \Lambda_v + \frac{c_{m,v} - c_{r,v}}{4} \tag{32}$$

We can finally determine the coordinates of the vertical tail tip leading edge with the same expressions that we used for the wing after flipping the y and z coordinates:

$$z_{t,v} = z_{r,v} + b_v \tag{33}$$

$$x_{t,v} = x_{r,v} + (z_{t,v} - z_{r,v}) \tan \Lambda_v + \frac{c_{r,v} - c_{t,v}}{4} \tag{34}$$

This concludes the sizing of the aerodynamic surfaces. You may have to adapt these equations if you intend to create unconventional configurations.

3.1.6 Test case

The script below shows how to use the `geoemtry` function for an aircraft based on the Fokker 100. Use it as a baseline for your new codes:

```python
# Sample script on how to use designTool.
# Remember to save this script in the same directory as designTool.py

# IMPORTS
import designTool as dt
import numpy
import pprint

# Inputs of the Geometry Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_w': 93.5,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_crew': 4463.55,
 'W_payload': 95519.97,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_slat_b_wing': 0.75,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_payload': 14.4,
 'xr_w': 13.5,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0}

# Execute the geometry function
dt.geometry(airplane)
```

```python

# Print updated dictionary
print('airplane = ' + pprint.pformat(airplane))

# Generate 3D plot
dt.plot3d(airplane)
```

If you implement the code correctly, you should get the following results after executing this script:

```
Outputs of the Geometry Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_crew': 4463.55,
 'W_payload': 95519.97,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_v': 4.358807176281144,
 'b_w': 28.074988869098416,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'cm_h': 2.107457619636192,
 'cm_v': 3.4576757510555542,
 'cm_w': 3.756317488774531,
 'cr_h': 2.849393124273043,
 'cr_v': 3.944978890651773,
 'cr_w': 5.3933059334262,
 'ct_h': 1.1112633184664868,
 'ct_v': 2.919284379082312,
 'ct_w': 1.267426894355157,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
'n_engines': 2,
'n_engines_under_wing': 0,
'range_altcruise': 370400.0,
'range_cruise': 2222400.0,
'rho_f': 804,
'slat_type': 'slat',
'sweep_h': 0.4537856055185257,
'sweep_v': 0.715584993317675,
'sweep_w': 0.3045599544730105,
'taper_h': 0.39,
'taper_v': 0.74,
'taper_w': 0.235,
'tcr_h': 0.1,
'tcr_v': 0.1,
'tcr_w': 0.123,
'tct_h': 0.1,
'tct_v': 0.1,
'tct_w': 0.096,
'x_mlg': 17.8,
'x_n': 23.2,
'x_nlg': 3.6,
'x_tailstrike': 23.68,
'x_tank_c_w': 0.2,
'xcg_crew': 2.5,
'xcg_payload': 14.4,
'xm_h': 34.21520026085125,
'xm_v': 31.17587613521955,
'xm_w': 15.659971822785682,
'xr_h': 33.07320337042791,
'xr_v': 29.25388711043971,
'xr_w': 13.5,
'xt_h': 35.74855563619494,
'xt_v': 33.299364009371466,
'xt_w': 18.944010614572072,
'y_mlg': 2.47,
'y_n': 2.6,
'ym_h': 1.9611423076663264,
'ym_w': 5.569532204800901,
'yt_h': 4.594361473577855,
'yt_w': 14.037494434549208,
'z_lg': -2.0,
'z_n': 0.0,
'z_tailstrike': -0.84,
'zm_h': 4.42748459846653,
'zm_v': 2.070850918999471,
'zm_w': 0.4872709290626237,
'zr_h': 4.359,
'zr_v': 0.0,
'zr_w': 0.0,
'zt_h': 4.519438637980579,
'zt_v': 4.358807176281144,
'zt_w': 1.2281216273313065}
```

You should also get the plot shown in Fig. 2:

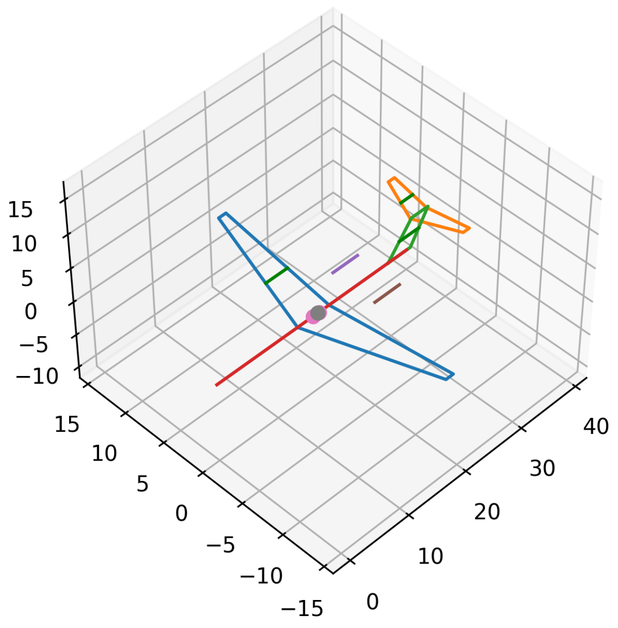


Figure 2: 3D view of the test case aircraft.

3.2 Aerodynamics module

The aerodynamics module is responsible for taking the three-dimensional description of the aircraft to estimate a drag polar of the form:

$$C_D = C_{D0} + K \cdot C_L^2 \tag{35}$$

This module should also estimate C_{Lmax} based on the 2D lift coefficient of the airfoil and flap settings.

The parasite drag will be estimated based on a component-based drag build-up:

$$C_{D0} = \sum_i C_{f,i} \cdot FF_i \cdot Q_i \cdot \frac{S_{wet,i}}{S_{ref}}, \tag{36}$$

where i indicates the component index, $C_{f,i}$ is the flat plate friction coefficient based on the component length, FF_i is the component form factor, Q_i is the component interference factor, $S_{wet,i}$ is the wetted area of the component, and S_{ref} is the reference area of the aircraft (usually the wing planform area).

Finally, the induced drag is be based on the Oswald factor (e):

$$K = \frac{1}{\pi \cdot AR_w \cdot e}. \tag{37}$$

3.2.1 Inputs

The dictionary inputs of this module are:

- Wing geometric parameters: S_w (`S_w`), AR_w (`AR_w`), $c_{r,w}$ (`cr_w`), $c_{t,w}$ (`ct_w`), λ_w (`taper_w`), Λ_w (`sweep_w`), $(t/c)_{r,w}$ (`tcr_w`), $(t/c)_{t,w}$ (`tct_w`), b_w (`b_w`), and $c_{m,w}$ (`cm_w`).
- Wing airfoil parameters: $c_{lmax,w}$ (`clmax_w`).
- Horizontal tail geometric parameters: S_h (`S_h`), $c_{r,h}$ (`cr_h`), $c_{t,h}$ (`ct_h`), λ_h (`taper_h`), Λ_h (`sweep_h`), $(t/c)_{r,h}$ (`tcr_h`), $(t/c)_{t,h}$ (`tct_h`), b_h (`b_h`), and $c_{m,h}$ (`cm_h`).
- Vertical tail geometric parameters: S_v (`S_v`), $c_{r,v}$ (`cr_v`), $c_{t,v}$ (`ct_v`), λ_v (`taper_v`), Λ_v (`sweep_v`), $(t/c)_{r,v}$ (`tcr_v`), $(t/c)_{t,v}$ (`tct_v`), b_v (`b_v`), and $c_{m,v}$ (`cm_v`).
- Fuselage parameters: L_f (`L_f`), and D_f (`D_f`).
- Nacelle parameters: L_n (`L_n`), and D_n (`D_n`).
- Engine parameters: n_{eng} (`n_engines`), and $n_{eng,w}$ (`n_engines_under_wing`).

- Flap parameters: δ_{flap} (flap_def), $\delta_{flap,TO}$ (TO_flap_def), $\delta_{flap,LD}$ (LD_flap_def), flap_type, c_{flap}/c_w (c_flap_c_wing), and b_{flap}/b_w (b_flap_b_wing).
- Slat parameters: δ_{slat} (slat_def), $\delta_{slat,TO}$ (TO_slat_def), $\delta_{slat,LD}$ (LD_slat_def), slat_type, c_{slat}/c_w (c_slat_c_wing), and b_{slat}/b_w (b_slat_b_wing).
- Excrescence drag parameter: k_{exc} (k_exc_drag).

The function also has additional input arguments to consider changes in the aircraft condition:

- Flight condition parameters: M (Mach), and h (altitude).
- Engine parameters: $n_{eng,f}$ (n_engines_failed).
- Flap parameters: δ_{flap} (flap_def).
- Slat parameters: δ_{slat} (slat_def).
- Landing gear parameter: lg_down.
- Ground effect parameter: h_{ground} (h_ground).
- Weight parameter: W_0 (W0_guess).

3.2.2 Outputs

The dictionary outputs of this module are:

- Fuselage parameters: $S_{wet,f}$ (Swet_f).

The additional outputs returned by this module are:

- Drag polar: C_{D0} (CD0), K (K), and C_{Lmax} (CLmax).

3.2.3 Parasite drag coefficient

The wetted areas are computed according to Torenbeek [1] Appendix B.

First we compute the exposed wing planform area (with respect to the fuselage) with:

$$S_{exp,w} = S_w - c_{r,w} \cdot D_f \tag{38}$$

Then we compute the wing wetted area with:

$$S_{wet,w} = 2 \cdot S_{exp,w} \cdot \left(1 + \frac{(t/c)_{r,w}}{4 \cdot (1+\lambda_w)} \cdot \left(1 + \lambda_w \cdot \frac{(t/c)_{r,w}}{(t/c)_{t,w}}\right)\right) \tag{39}$$

We use the auxiliary function Cf_calc to obtain the friction coefficient for the wing considering transition at 5%:

```
Cf_w = Cf_calc(Mach, altitude, length = cm_w, rugosity = rugosity, k_lam = 0.05)
```

Check Sec. 3.2.8 for a full description of this function. The rugosity parameter for smooth paint provided by Raymer (Tab 12.5) is $0.634 \cdot 10^{-5}$ meters. This value is used for all surfaces in this example.

We also use the auxiliary function FF_surface to compute the wing form factor:

```
FF_w = FF_surface(Mach, tcr_w, tct_w, sweep_w, b_w, cr_w, ct_w, cm_w)
```

Check Sec. 3.2.9 for a full description of this function. The interference factor for the wing can be neglected if we consider the use of fairings:

$$Q_w = 1.0 \tag{40}$$

We can finally compute the wing parasite drag $C_{D0,w}$ with:

$$C_{D0,w} = C_{f,w} \cdot FF_w \cdot Q_w \cdot \frac{S_{wet,w}}{S_w}, \tag{41}$$

Now we repeat the process for the horizontal tail. This time, we can consider the entire planform area of the tail as exposed area.

$$S_{exp,h} = S_h \tag{42}$$

Then we compute the horizontal tail wetted area with:

$$S_{wet,h} = 2 \cdot S_{exp,h} \cdot \left(1 + \frac{(t/c)_{r,h}}{4 \cdot (1 + \lambda_h)} \cdot \left(1 + \lambda_h \cdot \frac{(t/c)_{r,h}}{(t/c)_{t,h}}\right)\right) \tag{43}$$

We use the auxiliary function `Cf_calc` to obtain the friction coefficient for the horizontal tail considering transition at 5%:

```
Cf_h = Cf_calc(Mach, altitude, length = cm_h, rugosity = rugosity, k_lam = 0.05)
```

We also use the auxiliary function `FF_surface` to compute the horizontal tail form factor:

```
FF_h = FF_surface(Mach, tcr_h, tct_h, sweep_h, b_h, cr_h, ct_h, cm_h)
```

The interference factor for the horizontal tail can be neglected if we consider the use of fairings:

$$Q_h = 1.0 \tag{44}$$

We can finally compute the horizontal tail parasite drag $C_{D0,h}$ with:

$$C_{D0,h} = C_{f,h} \cdot FF_h \cdot Q_h \cdot \frac{S_{wet,h}}{S_w}, \tag{45}$$

We do this one more time for the vertical tail. The exposed area is:

$$S_{exp,v} = S_v \tag{46}$$

Then we compute the vertical tail wetted area with:

$$S_{wet,v} = 2 \cdot S_{exp,v} \cdot \left(1 + \frac{(t/c)_{r,v}}{4 \cdot (1 + \lambda_v)} \cdot \left(1 + \lambda_v \cdot \frac{(t/c)_{r,v}}{(t/c)_{t,v}}\right)\right) \tag{47}$$

We use the auxiliary function `Cf_calc` to obtain the friction coefficient for the vertical tail considering transition at 5%:

```
Cf_v = Cf_calc(Mach, altitude, length = cm_v, rugosity = rugosity, k_lam = 0.05)
```

We also use the auxiliary function `FF_surface` to compute the vertical tail form factor. Note that we have to double the span because the auxiliary function considers mirrored surfaces to compute sweep angles:

```
FF_v = FF_surface(Mach, tcr_v, tct_v, sweep_v, 2*b_v, cr_v, ct_v, cm_v)
```

The interference factor for the vertical tail can be neglected if we consider the use of fairings:

$$Q_v = 1.0 \tag{48}$$

We can finally compute the vertical tail parasite drag $C_{D0,v}$ with:

$$C_{D0,v} = C_{f,v} \cdot FF_v \cdot Q_v \cdot \frac{S_{wet,v}}{S_w}, \tag{49}$$

The next parameter we should compute is the fuselage fineness ratio:

$$\lambda_f = \frac{L_f}{D_f} \tag{50}$$

The fuselage wetted area is:

$$S_{wet,f} = \pi \cdot D_f \cdot L_f \cdot \left(1 - \frac{2}{\lambda_f}\right)^{2/3} \cdot \left(1 + \frac{1}{\lambda_f^2}\right) \tag{51}$$

We use the auxiliary function `Cf_calc` to obtain the friction coefficient for the fuselage considering transition at 5%:

```
Cf_f = Cf_calc(Mach, altitude, length = L_f, rugosity = rugosity, k_lam = 0.05)
```

Next we compute the fuselage form factor:

$$FF_f = 1 + \frac{60}{\lambda_f^3} + \frac{\lambda_f}{400} \tag{52}$$

There is no application of interference factor for the fuselage:

$$Q_f = 1.0 \tag{53}$$

We can finally compute the fuselage parasite drag $C_{D0,f}$ with:

$$C_{D0,f} = C_{f,f} \cdot FF_f \cdot Q_f \cdot \frac{S_{wet,f}}{S_w}, \tag{54}$$

We can estimate the wetted area of all nacelles with:

$$S_{wet,n} = n_{eng} \cdot \pi \cdot D_n \cdot L_n \tag{55}$$

We use the auxiliary function `Cf_calc` to obtain the friction coefficient for the nacelles considering transition at 5%:

```
Cf_n = Cf_calc(Mach, altitude, length = L_n, rugosity = rugosity, k_lam = 0.05)
```

Next we compute the nacelle form factor:

$$FF_n = 1 + 0.35 \cdot \frac{D_n}{L_n} \tag{56}$$

We apply an interference factor assuming that the distance between the nacelle and the wing or fuselage is less than its diameter:

$$Q_n = 1.2 \tag{57}$$

We can finally compute the nacelle parasite drag $C_{D0,n}$ with:

$$C_{D0,n} = C_{f,n} \cdot FF_n \cdot Q_n \cdot \frac{S_{wet,n}}{S_w}, \tag{58}$$

The total parasite drag will be:

$$C_{D0} = C_{D0,w} + C_{D0,h} + C_{D0,v} + C_{D0,f} + C_{D0,n} \tag{59}$$

The `aerodynamics` function also has a simpler drag estimation method where we apply a single friction coefficient to the entire wetted area of the airplane instead of adding individual drag coefficients. You can use this method by setting the optional argument `method=1` when calling the `aerodynamics` function.

In this simplified method, we start by computing the total wetted area:

$$S_{wet} = S_{wet,w} + S_{wet,h} + S_{wet,v} + S_{wet,f} + S_{wet,n} \tag{60}$$

The next steps will help us compute the equivalent friction coefficient for the entire airplane (C_{fe}). If we choose the wing area as our reference area ($S_{ref} = S_w$), then the area ratio is:

$$S_r = \frac{S_{wet}}{S_w} \tag{61}$$

The average thickness of the wing is:

$$(t/c)_w = \frac{(t/c)_{r,w} + (t/c)_{t,w}}{2} \tag{62}$$

The thickness correction factor (τ) is (Howe [2] Eq. 6.13b):

$$\tau = \frac{S_r - 2}{S_r} + \frac{1.9}{S_r} \cdot \left(1 + 0.526 \cdot (4 \cdot (t/c)_w)^3\right) \tag{63}$$

The friction coefficient can be computed with (Howe [2] Eq. 6.13a):

$$C_{fe} = 0.005 \cdot \left(1 - 2\frac{c_{lam}}{S_r}\right) \cdot \tau \cdot \left(1 - 0.2 \cdot M + 0.12 \cdot \left(\frac{M \cdot \sqrt{\cos \Lambda_w}}{A_f - (t/c)_w}\right)^{20}\right) \cdot T_f \cdot S_w^{-0.1} \tag{64}$$

For jet transport we can use (Howe [2] Tab. 6.4):

- airfoil factor: $A_f = 0.93$
- shape deviation factor: $T_f = 1.1$
- laminar flow fraction: $c_{lam} = 0.05$

The parasite drag coefficient of this simplified method can be finally computed as:

$$C_{D0} = C_{fe} \cdot S_r \tag{65}$$

3.2.4 Induced drag factor

We can now focus on the computation of the induced drag factor. First we need to compute a taper parameter:

$$f_\lambda = 0.005 \cdot (1 + 1.5 \cdot (\lambda_w - 0.6)^2) \tag{66}$$

then the Oswald factor is (Howe [2] Eq. 6.14):

$$e = \frac{1}{(1 + 0.12 \cdot M^6) \cdot \left(1 + \dfrac{0.142 + f_\lambda \cdot AR_w \cdot (10 \cdot (t/c)_w)^{0.33}}{(\cos \Lambda_w)^2} + 0.1 \cdot \dfrac{3 \cdot n_{eng,w} + 1}{(4 + AR_w)^{0.8}}\right)} \tag{67}$$

And the induced drag factor is:

$$K = \frac{1}{\pi \cdot AR_w \cdot e} \tag{68}$$

The ground effect may change the induced drag factor. For our code, we will assume that we should consider the ground effect only if $h_{ground} > 0$. If $h_{ground} > 0$ we should compute:

$$GE = 33 \cdot \left(\frac{h_{ground}}{b_w}\right)^{1.5} \tag{69}$$

$$K_{GE} = \frac{GE}{1 + GE} \tag{70}$$

Then we replace the induced drag factor value with (Raymer [3] Eq. 12.61):

$$K = K \cdot K_{GE} \tag{71}$$

3.2.5 Maximum lift coefficient and high-lift devices

For now, let's estimate the clean wing maximum lift coefficient with (Raymer Eq. 5.7):

$$C_{Lmax,clean} = 0.9 \cdot c_{lmax,w} \cdot \cos \Lambda_w \tag{72}$$

where $c_{lmax,w}$ is the 2D maximum lift coefficient of the wing airfoil.

The next steps will compute the flap and slats contributions to the maximum lift coefficient and parasite drag coefficient. Note that these equations will only be necessary if $\delta_{flap,max} > 0$ and $\delta_{slat,max} > 0$.

First we recompute the wing tip chord with:

$$c_{t,w} = \lambda_w \cdot c_{r,w} \tag{73}$$

The flap contribution to the parasite drag is (Raymer [3] Eq. 12.37):

$$C_{D0,flap} = 0.0023 \cdot (b_{flap}/b_w) \cdot \frac{180 \cdot \delta_{flap}}{\pi} \tag{74}$$

for δ_{flap} in radians.

Now we need to compute the wing sweep at the chord fraction corresponding to the flap hinge (which is located at $2 - c_{flap}/c_w$). You can use the auxiliary function `geo_change_sweep` from the `designTools.py` file for this purpose. To convert the sweep at the quarter-chord (Λ_w) to the sweep at the flap hinge we can use:

```
sweep_flap = geo_change_sweep(0.25, 2-c_flap_c_wing, sweep_w, b_w/2, cr_w, ct_w)
```

Then we can compute the lift increment with (Raymer [3] Eq. 12.21):

$$\Delta C_{Lmax,flap} = \Delta c_{lmax,flap} \cdot (b_{flap}/b_w) \cdot \cos \Lambda_{flap} \cdot \frac{\delta_{flap}}{\delta_{flap,max}} \tag{75}$$

The 2D lift increment depends on the flap type:

- plain flap: $\Delta c_{lmax,flap} = 0.9$
- slotted flap: $\Delta c_{lmax,flap} = 1.3$
- Fowler flap: $\Delta c_{lmax,flap} = 1.3 \cdot c_{flap}/c_w$
- double-slotted flap: $\Delta c_{lmax,flap} = 1.6 \cdot c_{flap}/c_w$
- triple-slotted flap: $\Delta c_{lmax,flap} = 1.9 \cdot c_{flap}/c_w$

We follow a similar procedure for leading edge devices (which we will generally refer as slats).
The slat contribution to the parasite drag is (Raymer [3] Eq. 12.37):

$$C_{D0,slat} = 0.0023 \cdot (b_{slat}/b_w) \cdot \frac{180 \cdot \delta_{slat}}{\pi} \tag{76}$$

for δ_{slat} in radians.

Now we need to compute the wing sweep at the chord fraction corresponding to the slat hinge (which is located at $c_{slat}/c_w - 1$). You can use the auxiliary function (`geo_change_sweep`) for this purpose. To convert the sweep at the quarter-chord (Λ_w) to the sweep at the slat hinge we can use:

```
sweep_slat = geo_change_sweep(0.25, c_slat_c_wing-1, sweep_w, b_w/2, cr_w, ct_w)
```

Then we can compute the lift increment with (Raymer [3] Eq. 12.21):

$$\Delta C_{Lmax,slat} = \Delta c_{lmax,slat} \cdot (b_{slat}/b_w) \cdot \cos \Lambda_{slat} \cdot \frac{\delta_{slat}}{\delta_{slat,max}} \tag{77}$$

The 2D lift increment depends on the leading edge device type:

- slot (fixed): $\Delta c_{lmax,slat} = 0.2$
- leading edge flap: $\Delta c_{lmax,slat} = 0.3$
- Kruger flap: $\Delta c_{lmax,slat} = 0.3$
- moving slat: $\Delta c_{lmax,slat} = 0.4 \cdot c_{slat}/c_w$

The maximum lift coefficient can be computed as:

$$C_{Lmax} = C_{Lmax,clean} + \Delta C_{Lmax,flap} + \Delta C_{Lmax,slat} \tag{78}$$

And the parasite drag coefficient can be updated as:

$$C_{D0} = C_{D0} + C_{D0,flap} + C_{D0,slat} \tag{79}$$

3.2.6 Additional components and excrescence drag

If the landing gear is deployed (`lg_down = 1`), we can compute its parasite drag contribution with (ESDU):

$$C_{D0,lg} = 0.001 \cdot \left(0.57 - 0.26 \cdot \frac{\delta_{flap}}{\delta_{flap,max}}\right) \cdot \left(\frac{W_0}{g}\right)^{0.785} \cdot \frac{1}{S_w} \tag{80}$$

where g is the acceleration of gravity. There is no need to compute this contribution if the landing gear is retracted (`lg_down = 0`).

If we have inoperative engines, we can compute the drag increase due to the windmilling effect (Raymer [3] Eq. 12.41).

$$C_{D0,windmill} = n_{eng,f} \cdot 0.3 \cdot \frac{\pi}{4} \cdot \frac{D_n^2}{S_w} \tag{81}$$

We can once again update the parasite drag coefficient with:

$$C_{D0} = C_{D0} + C_{D0,lg} + C_{D0,windmill} \tag{82}$$

We can finally apply an excrescence drag factor to account for leaks and protuberances:

$$C_{D0} = \frac{C_{D0}}{1 - k_{exc}} \tag{83}$$

the k_{exc} is usually between 0.03 and 0.06.

3.2.7 Wave drag

The appearance of shock waves at transonic speeds increases drag. The expressions below estimate this drag contribution, but they should ONLY be used if $M > 0.4$. Otherwise, assume that there is no wave drag ($C_{D,wave} = 0$), what can be considered reasonable according to Fig. 3. This check is necessary because the equations below will estimate a spurious drag for very low Mach numbers.

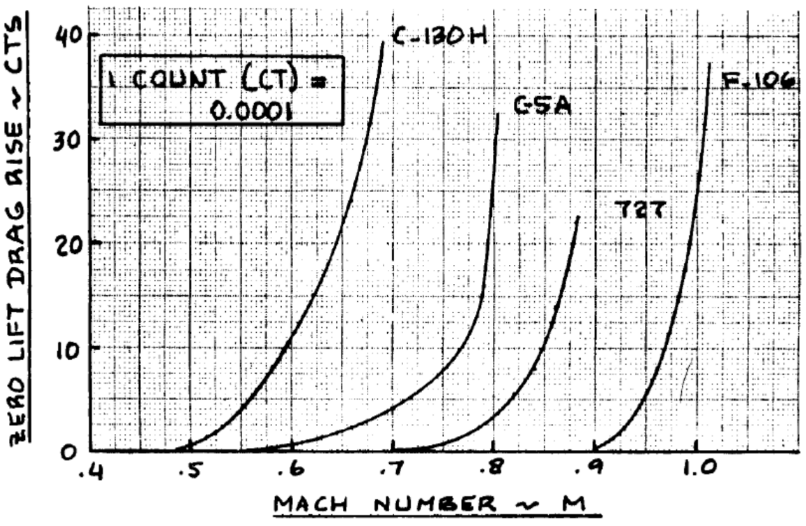


Figure 3: Wave drag for different aircraft (Roskam [4] Fig. 12.7).

We will adapt the Korn equation to estimate wave drag. We will assume that the flight C_L should satisfy $L = W$. First use the function `atmosphere` provided in the auxiliary files to compute the air properties at the flight altitude:

```
T,p,rho,mi = atmosphere(altitude, 288.15)
```

Use the temperature to compute the speed of sound with:

$$a = \sqrt{\gamma \cdot R \cdot T} \tag{84}$$

for air we have $\gamma = 1.4$ and $R = 287$. We can find the flight speed with:

$$V = M \cdot a \tag{85}$$

Now we estimate the lift coefficient based on the $L = W_0$ relationship:

$$C_L = \frac{2 \cdot W_0}{\rho \cdot V^2 \cdot S_w} \tag{86}$$

Then we compute the divergence Mach number with:

$$M_{dd} = \frac{0.95}{\cos \Lambda_w} - \frac{(t/c)_w}{(\cos \Lambda_w)^2} - \frac{C_L}{10 \cdot (\cos \Lambda_w)^3} \tag{87}$$

where $(t/c)_w$ is the average wing thickness given by Eq. (62). The critical Mach number is:

$$M_c = M_{dd} - \left(\frac{0.1}{80}\right)^{1/3} \tag{88}$$

If the flight Mach number is beyond the critical Mach number (that is, $M > M_c$):

$$C_{D,wave} = 20 \cdot (M - M_c)^4 \tag{89}$$

otherwise:

$$C_{D,wave} = 0 \tag{90}$$

For now, we can embed the wave drag into the parasite drag coefficient:

$$C_{D0} = C_{D0} + C_{D,wave} \tag{91}$$

Note that we enforced $L = W_0$ to compute the C_L used in Eq. 87, but the Korn equation could be used for other C_L values.

3.2.8 Friction coefficient

The (total) friction coefficient is computed by averaging empirical flat plate results according to laminar and turbulent fractions. Then we apply this friction coefficient to the entire wetted area of the component.

First we compute the Reynolds number using the entire reference length L of the component:

$$Re_{L,1} = \frac{\rho \cdot V \cdot L}{\mu} \tag{92}$$

where rho, V, and μ are the density, speed, and dynamic viscosity of the corresponding flow condition, respectively.

The surface rugosity may also influence the friction coefficient. Raymer suggests that we take this factor into account with a cutoff Reynolds number:

$$Re_{cutoff} = 38.21 \cdot \left(\frac{L}{r}\right)^{1.053} \tag{93}$$

The rugosity parameter r can be obtained from Fig. 4.

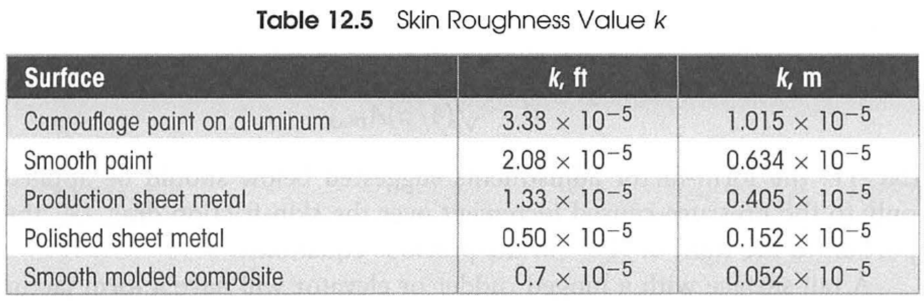


Table 12.5 Skin Roughness Value k

Surface	k, ft	k, m
Camouflage paint on aluminum	3.33×10^{-5}	1.015×10^{-5}
Smooth paint	2.08×10^{-5}	0.634×10^{-5}
Production sheet metal	1.33×10^{-5}	0.405×10^{-5}
Polished sheet metal	0.50×10^{-5}	0.152×10^{-5}
Smooth molded composite	0.7×10^{-5}	0.052×10^{-5}

Figure 4: Skin rugosity provided by Raymer [3] (Tab. 12.5). Use the k value as r in the derivations presented here.

We should use the smaller Reynolds number for the next computation. In other words:

$$Re_L = \min(Re_{L,1}, Re_{cutoff}) \tag{94}$$

We also need to compute the Reynolds number where transition happens. If we define k_{trans} as the fraction of reference length L where transition occurs, we have:

$$Re_{trans,1} = \frac{\rho \cdot V \cdot k_{trans} \cdot L}{\mu} \tag{95}$$

We also compute the cutoff Reynolds number at the transition:

$$Re_{trans,cutoff} = 38.21 \cdot \left(\frac{k_{trans} \cdot L}{r}\right)^{1.053} \tag{96}$$

then we take the smallest Reynolds number:

$$Re_{trans} = \min(Re_{trans,1}, Re_{cutoff}) \tag{97}$$

Next we compute the laminar friction coefficient at the transition point with:

$$C_{f,lam,trans} = \frac{1.328}{\sqrt{Re_{trans}}} \tag{98}$$

Then we compute the turbulent friction coefficient at the end of the reference length with:

$$C_{f,turb,L} = \frac{0.455}{(\log_{10} Re_L)^{2.58} \cdot (1 + 0.144 \cdot M^2)^{0.65}} \tag{99}$$

where M is the Mach number. We also need to determine the turbulent friction coefficient at the transition point:

$$C_{f,turb,trans} = \frac{0.455}{(\log_{10} Re_{trans})^{2.58} \cdot (1 + 0.144 \cdot M^2)^{0.65}} \tag{100}$$

We can finally compute the averaged friction coefficient with:

$$C_f = k_{trans} \cdot (C_{f,lam,trans} - C_{f,turb,trans}) + C_{f,turb,L} \tag{101}$$

These equations are implemented in the auxiliary function `Cf_calc`.

3.2.9 Form factor of lifting surfaces

The form factor of lifting surfaces is computed as;

$$FF = 1.34 \cdot M^{0.18} \cdot (\cos \Lambda_{max(t/c)})^{0.28} \cdot \left(1 + 0.6 \cdot \frac{(t/c)}{(x/c)_{max(t/c)} \cdot c_m} + 100 \cdot (t/c)^4\right) \tag{102}$$

where $\Lambda_{max(t/c)}$ is the sweep angle at the chord fraction of the maximum thickness position, (t/c) is the average thickness-to-chord ratio of the surface, $(x/c)_{max(t/c)}$ is the chord fraction of the maximum thickness position, and c_m is the mean aerodynamic chord of the surface.

This equation is implemented in the auxiliary function `FF_surface`.

3.2.10 Test case

Consider a test case with the following inputs:

```
Inputs of the Aerodynamics Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_crew': 4463.55,
 'W_payload': 95519.97,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
'b_h': 9.18872294715571,
'b_slat_b_wing': 0.75,
'b_v': 4.358807176281144,
'b_w': 28.074988869098416,
'c_flap_c_wing': 1.2,
'c_slat_c_wing': 1.05,
'c_tank_c_w': 0.4,
'clmax_w': 2.1,
'cm_h': 2.107457619636192,
'cm_v': 3.4576757510555542,
'cm_w': 3.756317488774531,
'cr_h': 2.849393124273043,
'cr_v': 3.944978890651773,
'cr_w': 5.3933059334262,
'ct_h': 1.1112633184664868,
'ct_v': 2.919284379082312,
'ct_w': 1.267426894355157,
'dihedral_h': 0.03490658503988659,
'dihedral_w': 0.08726646259971647,
'distance_landing': 1800.0,
'distance_takeoff': 1800.0,
'eta_h': 1.0,
'flap_type': 'double slotted',
'h_ground': 10.668000000000001,
'k_exc_drag': 0.03,
'loiter_time': 2700,
'n_engines': 2,
'n_engines_under_wing': 0,
'range_altcruise': 370400.0,
'range_cruise': 2222400.0,
'rho_f': 804,
'slat_type': 'slat',
'sweep_h': 0.4537856055185257,
'sweep_v': 0.715584993317675,
'sweep_w': 0.3045599544730105,
'taper_h': 0.39,
'taper_v': 0.74,
'taper_w': 0.235,
'tcr_h': 0.1,
'tcr_v': 0.1,
'tcr_w': 0.123,
'tct_h': 0.1,
'tct_v': 0.1,
'tct_w': 0.096,
'x_mlg': 17.8,
'x_n': 23.2,
'x_nlg': 3.6,
'x_tailstrike': 23.68,
'x_tank_c_w': 0.2,
'xcg_crew': 2.5,
'xcg_payload': 14.4,
'xm_h': 34.21520026085125,
'xm_v': 31.17587613521955,
'xm_w': 15.659971822785682,
'xr_h': 33.07320337042791,
'xr_v': 29.25388711043971,
'xr_w': 13.5,
'xt_h': 35.74855563619494,
'xt_v': 33.299364009371466,
'xt_w': 18.944010614572072,
'y_mlg': 2.47,
'y_n': 2.6,
'ym_h': 1.9611423076663264,
'ym_w': 5.569532204800901,
'yt_h': 4.594361473577855,
'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}

Mach = 0.30000000000000

altitude = 10.66800000000000

n_engines_failed = 1.00000000000000

flap_def = 0.34906585039887

slat_def = 0.00000000000000

lg_down = 1.00000000000000

h_ground = 10.66800000000000

W0_guess = 467500.00000000000000
```

You should get the following results:

```
Outputs of the Aerodynamics Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_crew': 4463.55,
 'W_payload': 95519.97,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_v': 4.358807176281144,
 'b_w': 28.074988869098416,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'cm_h': 2.107457619636192,
 'cm_v': 3.4576757510555542,
 'cm_w': 3.756317488774531,
 'cr_h': 2.849393124273043,
 'cr_v': 3.944978890651773,
 'cr_w': 5.3933059334262,
 'ct_h': 1.1112633184664868,
 'ct_v': 2.919284379082312,
 'ct_w': 1.267426894355157,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}

CD0 = 0.07154895353144

K = 0.04101373267785

CLmax = 2.37303456079851
```

3.2.11 Homework

Modify the following variables from the previous example:

```
altitude = 11000
n_engines_failed = 0
flap_def = 0.0
slat_def = 0.0
lg_down = 0
h_ground = 0.0
W0_guess = 41500*9.81

airplane['sweep_w'] = 10*np.pi/180
```

1. Generate a plot of C_{D0} as a function of M (Mach number) for the interval $0.6 < M < 0.8$. Then increase the wing sweep in intervals of 5 degrees until you reach 40 degrees and draw new drag curves on the same plot.

   ATTENTION: Remember to execute the *geometry* module whenever you change sweep to update all aircraft dimensions.

2. Compute C_{Lmax} for the same sweep angles described above and generate a plot of C_{Lmax} as a function of wing sweep angle.

3. Based on the two previous plots, describe the advantages and disadvantages of increasing the sweep angle of the wing. Indicate which flight phases are favored and which ones are unfavored by high values of sweep angles.

4. Reset the reference values from the test case and compute drag polar coefficients for the following conditions:

   - Cruise:

     ```
     Mach = 0.75
     altitude = 11000
     n_engines_failed = 0
     flap_def = 0.0
     slat_def = 0.0
     lg_down = 0
     h_ground = 0
     W0_guess = 41500*9.81
     ```

   - Takeoff:

     ```
     Mach = 0.2
     altitude = 0
     n_engines_failed = 0
     flap_def = 20*np.pi/180
     slat_def = 0.0
     lg_down = 1
     h_ground = 10.67
     W0_guess = 41500*9.81
     ```

- Landing:

  ```
  Mach = 0.2
  altitude = 0
  n_engines_failed = 0
  flap_def = 40*np.pi/180
  slat_def = 0.0
  lg_down = 1
  h_ground = 10.67
  W0_guess = 38300*9.81
  ```

  Use the coefficients to draw drag polars for these three configurations on the same plot. You can use $C_{Lmin} = -0.5$ as the lower bound for the lift coefficient in this plot.

5. A preliminary performance calculation showed that the takeoff requires a lift coefficient of 2.1. The following high-lift device options are available:

   - Plain flap:

     ```
     flap_type = 'plain'
     flap_def = 20*np.pi/180
     max_flap_def = 60*np.pi/180
     ```

   - Single slotted flap:

     ```
     flap_type = 'slotted'
     flap_def = 20*np.pi/180
     max_flap_def = 40*np.pi/180
     ```

   - Fowler flap:

     ```
     flap_type = 'fowler'
     flap_def = 15*np.pi/180
     max_flap_def = 40*np.pi/180
     ```

   - Double slotted flap:

     ```
     flap_type = 'double slotted'
     flap_def = 20*np.pi/180
     max_flap_def = 50*np.pi/180
     ```

   - Triple slotted flap:

     ```
     flap_type = 'triple slotted'
     flap_def = 20*np.pi/180
     max_flap_def = 40*np.pi/180
     ```

   Use the same flight conditions defined for takeoff in the previous exercise to estimate aerodynamic coefficients for these configurations. Then select a high-lift device according to the requirement and justify your choice.

6. Repeat the same steps above for your aircraft. Use geometrical data from historical trends and fuselage dimensions obtained from the previous assignment. You can also obtain airfoil (t/c and c_{lmax}) and high-lift device information from similar aircraft for this initial analysis.

Summarize your results in a PDF document and send it to ney@ita.br with the title AP-701 HW02.

3.3 Specific fuel consumption module

We need a module that computes the specific fuel consumption (C) of the engines according to their operating conditions.

3.3.1 Inputs

The dictionary inputs for this module are:

- Engine parameter: BPR (`BPR`).
- Base specific fuel consumption of the engine (optional): C_{base} (`Cbase`).

The function also has additional arguments to consider different flight conditions:

- Flight condition parameters: M (`Mach`), and h (`altitude`).

3.3.2 Outputs

This function does not modify the dictionary.

The outputs returned by this module are:

- Engine parameter: C (`C`).

3.3.3 Specific fuel consumption

First use the auxiliary function provided in the `designTools.py` file to get the atmospheric properties:

```
T,p,rho,mi = atmosphere(altitude, 288.15)
```

Then compute the ratio σ between the current air density and the sea-level air density (which is 1.225 kg/m^3):

$$\sigma = \frac{\rho}{1.225} \tag{103}$$

Determine the base specific fuel consumption according to the engine type:

- Low bypass ratio engines ($BPR < 4.0$): $C_{base} = 0.85/3600$ 1/s
- High bypass ratio engines ($BPR \geq 4.0$): $C_{base} = 0.70/3600$ 1/s

Now compute the specific fuel consumption at the current flight condition with (Howe [2] Eq. 3.12a):

$$C = C_{base} \cdot \left(1 - 0.15 \cdot BPR^{0.65}\right) \cdot \left(1 + 0.28 \cdot \left(1 + 0.063 \cdot BPR^2\right) \cdot M\right) \cdot \sigma^{0.08} \tag{104}$$

3.3.4 Test case

Consider a test case with the following inputs:

```
Inputs of the TSFC Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_crew': 4463.55,
 'W_payload': 95519.97,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_v': 4.358807176281144,
 ’b_w’: 28.074988869098416,
 ’c_flap_c_wing’: 1.2,
 ’c_slat_c_wing’: 1.05,
 ’c_tank_c_w’: 0.4,
 ’clmax_w’: 2.1,
 ’cm_h’: 2.107457619636192,
 ’cm_v’: 3.4576757510555542,
 ’cm_w’: 3.756317488774531,
 ’cr_h’: 2.849393124273043,
 ’cr_v’: 3.944978890651773,
 ’cr_w’: 5.3933059334262,
 ’ct_h’: 1.1112633184664868,
 ’ct_v’: 2.919284379082312,
 ’ct_w’: 1.267426894355157,
 ’dihedral_h’: 0.03490658503988659,
 ’dihedral_w’: 0.08726646259971647,
 ’distance_landing’: 1800.0,
 ’distance_takeoff’: 1800.0,
 ’eta_h’: 1.0,
 ’flap_type’: ’double slotted’,
 ’h_ground’: 10.668000000000001,
 ’k_exc_drag’: 0.03,
 ’loiter_time’: 2700,
 ’n_engines’: 2,
 ’n_engines_under_wing’: 0,
 ’range_altcruise’: 370400.0,
 ’range_cruise’: 2222400.0,
 ’rho_f’: 804,
 ’slat_type’: ’slat’,
 ’sweep_h’: 0.4537856055185257,
 ’sweep_v’: 0.715584993317675,
 ’sweep_w’: 0.3045599544730105,
 ’taper_h’: 0.39,
 ’taper_v’: 0.74,
 ’taper_w’: 0.235,
 ’tcr_h’: 0.1,
 ’tcr_v’: 0.1,
 ’tcr_w’: 0.123,
 ’tct_h’: 0.1,
 ’tct_v’: 0.1,
 ’tct_w’: 0.096,
 ’x_mlg’: 17.8,
 ’x_n’: 23.2,
 ’x_nlg’: 3.6,
 ’x_tailstrike’: 23.68,
 ’x_tank_c_w’: 0.2,
 ’xcg_crew’: 2.5,
 ’xcg_payload’: 14.4,
 ’xm_h’: 34.21520026085125,
 ’xm_v’: 31.17587613521955,
 ’xm_w’: 15.659971822785682,
 ’xr_h’: 33.07320337042791,
 ’xr_v’: 29.25388711043971,
 ’xr_w’: 13.5,
 ’xt_h’: 35.74855563619494,
 ’xt_v’: 33.299364009371466,
 ’xt_w’: 18.944010614572072,
 ’y_mlg’: 2.47,
 ’y_n’: 2.6,
 ’ym_h’: 1.9611423076663264,
 ’ym_w’: 5.569532204800901,
 ’yt_h’: 4.594361473577855,
 ’yt_w’: 14.037494434549208,
 ’z_lg’: -2.0,
 ’z_n’: 0.0,
 ’z_tailstrike’: -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}

altitude = 10668.00000000000000

Mach = 0.73000000000000
```

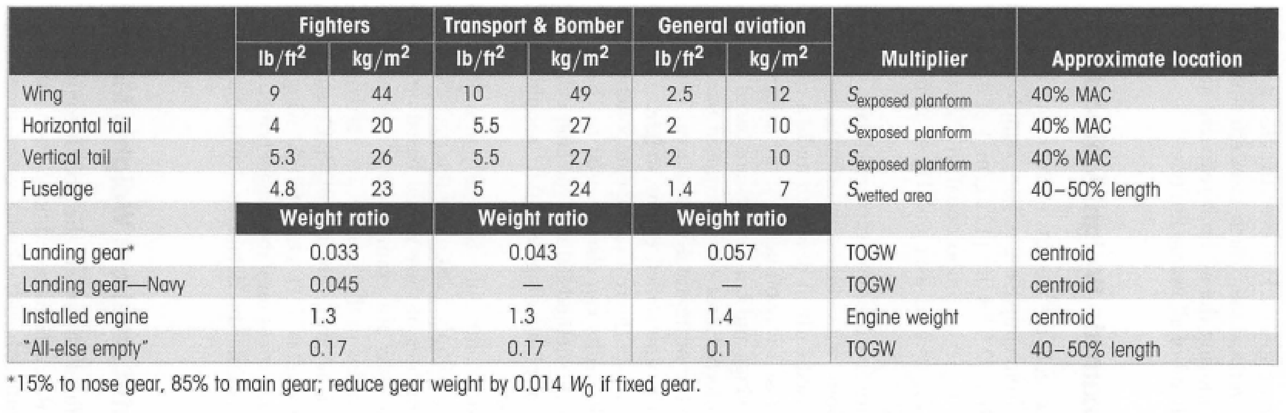
You should get the following results:

```
Outputs of the TSFC Module

C = 0.00019663493654
```

3.4 Empty weight module

This module estimates the empty weight of the airplane by adding contributions from its components. We will use some historical trends from Fig. 5 for this purpose.



	Fighters		Transport & Bomber		General aviation		Multiplier	Approximate location
	lb/ft^2	kg/m^2	lb/ft^2	kg/m^2	lb/ft^2	kg/m^2		
Wing	9	44	10	49	2.5	12	$S_{\text{exposed planform}}$	40% MAC
Horizontal tail	4	20	5.5	27	2	10	$S_{\text{exposed planform}}$	40% MAC
Vertical tail	5.3	26	5.5	27	2	10	$S_{\text{exposed planform}}$	40% MAC
Fuselage	4.8	23	5	24	1.4	7	$S_{\text{wetted area}}$	40–50% length
	Weight ratio		Weight ratio		Weight ratio			
Landing gear*	0.033		0.043		0.057		TOGW	centroid
Landing gear—Navy	0.045		—		—		TOGW	centroid
Installed engine	1.3		1.3		1.4		Engine weight	centroid
"All-else empty"	0.17		0.17		0.1		TOGW	40–50% length

*15% to nose gear, 85% to main gear; reduce gear weight by 0.014 W_0 if fixed gear.

Figure 5: Empty weight contributions estimated by Raymer [3] (Tab. 15.2).

We will use the same module to estimate the CG locations of the empty aircraft.

3.4.1 Inputs

The dictionary inputs for this module are:

- Wing parameters: S_w (`S_w`), AR_w (`AR_w`), λ_w (`taper_w`), Λ_w (`sweep_w`), $x_{m,w}$ (`xm_w`), $c_{m,w}$ (`cm_w`), and $(t/c)_{r,w}$ (`tcr_w`).
- Horizontal tail parameters: S_h (`S_h`), $x_{m,h}$ (`xm_h`), and $c_{m,h}$ (`cm_h`).
- Vertical tail parameters: S_v (`S_v`), $x_{m,v}$ (`xm_v`), and $c_{m,v}$ (`cm_v`).
- Fuselage parameters: L_f (`L_f`), and $S_{wet,f}$ (`Swet_f`).
- Engine parameters: n_{eng} (`n_engines`), BPR (`BPR`), x_n (`x_n`), and L_n (`L_n`).
- Landing gear parameters: x_{nlg} (`x_nlg`) and x_{mlg} (`x_mlg`).

The function also has additional input arguments to aid in iterative procedures

- Weight parameter: W_0 (`W0_guess`).
- Engine parameters: T_0 (`T0_guess`).

3.4.2 Outputs

The dictionary outputs of this module are:

- Weight parameter: W_w (`W_w`), W_h (`W_h`), W_v (`W_v`), W_f (`W_f`), W_{nlg} (`W_nlg`), W_{mlg} (`W_mlg`), W_{eng} (`W_eng_installed`), and W_{ae} (`W_allelse`).

The additional outputs returned by this module are (they are assigned to the dictionary later on):

- Weight parameter: W_e (`We`).
- CG parameter: $x_{CG,e}$ (`xcg_e`).

3.4.3 Wing weight

Compute the ultimate load factor with:

$$N_z = 1.5 \cdot 2.5 \tag{105}$$

Estimate the control surface area as 15% of the total wing area:

$$S_{csw} = 0.15 \cdot S_w \tag{106}$$

Now we can estimate the wing weight with the following expression (Raymer [3] Eq. 15.25):

$$W_w = 0.0051 \cdot (W_0 \cdot N_z)^{0.557} \cdot S_w^{0.649} \cdot AR_w^{0.55} \cdot (t/c)_{r,w}^{-0.4} \cdot (1+\lambda_w)^{0.1} \cdot (\cos\Lambda_w)^{-1.0} \cdot S_{csw}^{0.1} \tag{107}$$

ATTENTION: This equation uses British units! All areas are in ft^2 and all weights are in lbs. Remember to make conversions before and after using this equation.

Next we estimate the wing center of gravity according to Fig. 5:

$$x_{CG,w} = x_{m,w} + 0.4 \cdot c_{m,w} \tag{108}$$

NOTE: I increased the AR_w exponent from 0.5 to 0.55 compared to the original Eq. 15.25 from Raymer [3] since it was not penalizing high aspect ratios. For instance, a rough adaption of Torenbeek [1] Eq. C-1 suggests that this exponent should be closer to 0.8.

3.4.4 Horizontal tail weight

According to Fig. 5, we can compute the weight of this component with:

$$W_h = 27 \cdot g \cdot S_h \tag{109}$$

where g is the acceleration of gravity (9.81 m/s^2).

The center of gravity of this component, also estimated with Fig. 5, is given by:

$$x_{CG,h} = x_{m,h} + 0.4 \cdot c_{m,h} \tag{110}$$

3.4.5 Vertical tail weight

According to Fig. 5, we can compute the weight of this component with:

$$W_v = 27 \cdot g \cdot S_v \tag{111}$$

The center of gravity of this component, also estimated with Fig. 5, is given by:

$$x_{CG,v} = x_{m,v} + 0.4 \cdot c_{m,v} \tag{112}$$

3.4.6 Fuselage weight

According to Fig. 5, we can compute the weight of this component with:

$$W_f = 24 \cdot g \cdot S_{wet,f} \tag{113}$$

The center of gravity of this component, also estimated with Fig. 5, is given by:

$$x_{CG,f} = 0.45 \cdot L_f \tag{114}$$

3.4.7 Nose landing gear weight

According to Fig. 5, we can compute the weight of this component with:

$$W_{nlg} = 0.15 \cdot 0.043 \cdot W_0 \tag{115}$$

Remember to use `W0_guess` as W_0.
The center of gravity of this component, also estimated with Fig. 5, is given by:

$$x_{CG,nlg} = x_{nlg} \tag{116}$$

3.4.8 Main landing gear weight

According to Fig. 5, we can compute the weight of this component with:

$$W_{mlg} = 0.85 \cdot 0.043 \cdot W_0 \tag{117}$$

Remember to use `W0_guess` as W_0.
The center of gravity of this component, also estimated with Fig. 5, is given by:

$$x_{CG,mlg} = x_{mlg} \tag{118}$$

3.4.9 Installed engine weight

First determine the take-off thrust of a single engine with:

$$T_{eng,s} = \frac{T_0}{n_{eng}} \tag{119}$$

Remember to use `T0_guess` as T_0 in the equation above.
The isolated engine weight can be estimated with (Raymer [3] Eq. 10.4):

$$W_{eng,s} = 14.7 \cdot g \cdot \left(\frac{T_{eng}}{1000}\right)^{1.1} \cdot e^{-0.045 \cdot BPR} \tag{120}$$

Now the installed engine weight, according to Fig. 5, is given by:

$$W_{eng} = 1.3 \cdot n_{eng} \cdot W_{eng,s} \tag{121}$$

Figure 5 also lets us estimate the center of gravity position of the installed engines as:

$$x_{CG,eng} = x_n + 0.5 \cdot L_n \tag{122}$$

3.4.10 All-else weight

According to Fig. 5, the weight of the remaining systems and components can be calculated as:

$$W_{ae} = 0.17 \cdot W_0 \tag{123}$$

Remember to use `W0_guess` as W_0.
For now, lets compute the corresponding center of gravity position by using the average value suggested by Fig. 5:

$$x_{CG,ae} = 0.45 \cdot L_f \tag{124}$$

You may have freedom to adjust this parameter to get better static margins later in the design process.

3.4.11 Empty weight

The overall empty weight is the sum of the weights of all components:

$$W_e = W_w + W_h + W_v + W_f + W_{nlg} + W_{mlg} + W_{eng} + W_{ae} \tag{125}$$

The empty aircraft center of gravity is the weighted average of the center of gravity of all components:

$$x_{CG,e} = \frac{W_w x_{CG,w} + W_h x_{CG,h} + W_v x_{CG,v} + W_f x_{CG,f} + W_{nlg} x_{CG,nlg} + W_{mlg} x_{CG,mlg} + W_{eng} x_{CG,eng} + W_{ae} x_{CG,ae}}{W_e} \tag{126}$$

These are the two outputs of this module.

3.4.12 Test case

Consider a test case with the following inputs:

```
Inputs of the Empty Weight Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_crew': 4463.55,
 'W_payload': 95519.97,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_v': 4.358807176281144,
 'b_w': 28.074988869098416,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'cm_h': 2.107457619636192,
 'cm_v': 3.4576757510555542,
 'cm_w': 3.756317488774531,
 'cr_h': 2.849393124273043,
 'cr_v': 3.944978890651773,
 'cr_w': 5.3933059334262,
 'ct_h': 1.1112633184664868,
 'ct_v': 2.919284379082312,
 'ct_w': 1.267426894355157,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}

W0_guess = 467500.00000000000000

T0_guess = 140250.00000000000000
```

You should get the following results:

```
Outputs of the Empty Weight Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
'LD_slat_def': 0.0,
'L_f': 32.5,
'L_n': 4.3,
'Lb_v': 0.55,
'Lc_h': 4.83,
'MLW_frac': 0.9228915662650602,
'Mach_altcruise': 0.4,
'Mach_cruise': 0.73,
'S_h': 18.196687370600415,
'S_v': 14.959999999999999,
'S_w': 93.5,
'Swet_f': 292.60345689585,
'TO_flap_def': 0.3490658503988659,
'TO_slat_def': 0.0,
'W_allelse': 79475.0,
'W_crew': 4463.55,
'W_eng': 35075.863123799056,
'W_f': 68890.55789155893,
'W_h': 4819.756583850933,
'W_mlg': 17087.125,
'W_nlg': 3015.3749999999995,
'W_payload': 95519.97,
'W_v': 3962.4552,
'W_w': 34840.475533682125,
'altitude_altcruise': 4572,
'altitude_cruise': 10668.0,
'altitude_landing': 0.0,
'altitude_takeoff': 0.0,
'b_flap_b_wing': 0.6,
'b_h': 9.18872294715571,
'b_slat_b_wing': 0.75,
'b_v': 4.358807176281144,
'b_w': 28.074988869098416,
'c_flap_c_wing': 1.2,
'c_slat_c_wing': 1.05,
'c_tank_c_w': 0.4,
'clmax_w': 2.1,
'cm_h': 2.107457619636192,
'cm_v': 3.4576757510555542,
'cm_w': 3.756317488774531,
'cr_h': 2.849393124273043,
'cr_v': 3.944978890651773,
'cr_w': 5.3933059334262,
'ct_h': 1.1112633184664868,
'ct_v': 2.919284379082312,
'ct_w': 1.267426894355157,
'dihedral_h': 0.03490658503988659,
'dihedral_w': 0.08726646259971647,
'distance_landing': 1800.0,
'distance_takeoff': 1800.0,
'eta_h': 1.0,
'flap_type': 'double slotted',
'h_ground': 10.668000000000001,
'k_exc_drag': 0.03,
'loiter_time': 2700,
'n_engines': 2,
'n_engines_under_wing': 0,
'range_altcruise': 370400.0,
'range_cruise': 2222400.0,
'rho_f': 804,
'slat_type': 'slat',
'sweep_h': 0.4537856055185257,
'sweep_v': 0.715584993317675,
'sweep_w': 0.3045599544730105,
'taper_h': 0.39,
'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}

We = 247166.60833289104630

xcg_e = 17.27563652209663
```

3.5 Fuel weight module

This module computes the fuel contribution to the MTOW. Some of the fuel fractions can be estimated using the historical data shown in Fig. 6.

3.5.1 Inputs

The dictionary inputs of this module are:

- Wing parameter: S_w (`S_w`).
- Cruise parameters: h_{cruise} (`altitude_cruise`), M_{cruise} (`Mach_cruise`), R_{cruise} (`range_cruise`).
- Loiter parameter: E_{loiter} (`loiter_time`).
- Alternative cruise parameters: $h_{altcruise}$ (`altitude_altcruise`), $M_{altcruise}$ (`Mach_altcruise`), and $R_{altcruise}$ (`range_altcruise`).

Dictionary parameters used by **`aerodynamics`** and **`engineTSFC`** functions are also necessary.
The function also has additional input arguments to aid in iterative procedures:

- Weight parameter: W_0 (`W0_guess`).

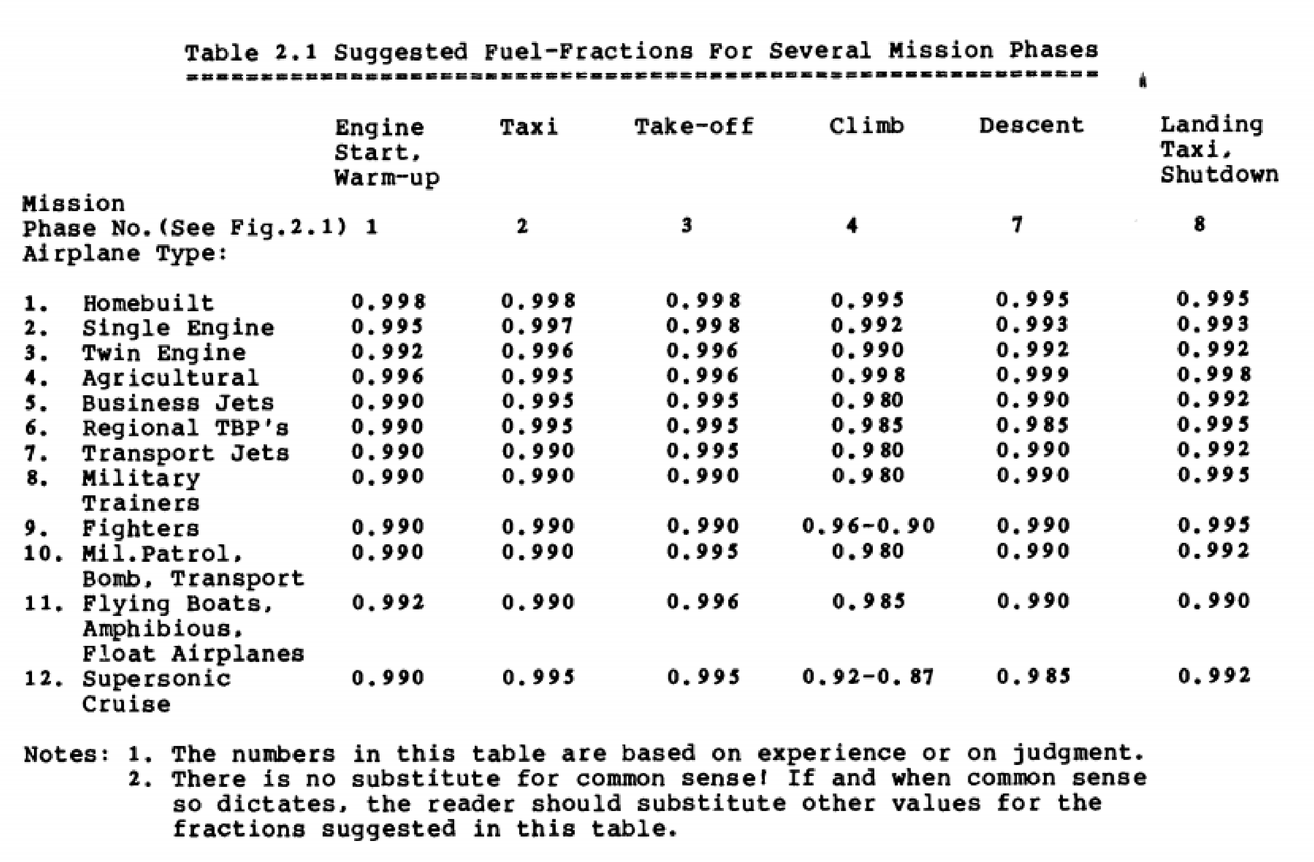


Table 2.1 Suggested Fuel-Fractions For Several Mission Phases

Mission Phase No. (See Fig. 2.1) / Airplane Type:	Engine Start, Warm-up	Taxi	Take-off	Climb	Descent	Landing Taxi, Shutdown
	1	2	3	4	7	8
1. Homebuilt	0.998	0.998	0.998	0.995	0.995	0.995
2. Single Engine	0.995	0.997	0.998	0.992	0.993	0.993
3. Twin Engine	0.992	0.996	0.996	0.990	0.992	0.992
4. Agricultural	0.996	0.995	0.996	0.998	0.999	0.998
5. Business Jets	0.990	0.995	0.995	0.980	0.990	0.992
6. Regional TBP's	0.990	0.995	0.995	0.985	0.985	0.995
7. Transport Jets	0.990	0.990	0.995	0.980	0.990	0.992
8. Military Trainers	0.990	0.990	0.990	0.980	0.990	0.995
9. Fighters	0.990	0.990	0.990	0.96-0.90	0.990	0.995
10. Mil.Patrol, Bomb, Transport	0.990	0.990	0.995	0.980	0.990	0.992
11. Flying Boats, Amphibious, Float Airplanes	0.992	0.990	0.996	0.985	0.990	0.990
12. Supersonic Cruise	0.990	0.995	0.995	0.92-0.87	0.985	0.992

Notes: 1. The numbers in this table are based on experience or on judgment.
2. There is no substitute for common sense! If and when common sense so dictates, the reader should substitute other values for the fractions suggested in this table.

Figure 6: Fuel fractions estimated by Roskam [5].

3.5.2 Outputs

The outputs of this module are:

- Weight parameters: W_f (`Wf`) and $M_{f,cruise}$ (`Mf_cruise`).

This function does not modify the dictionary

3.5.3 Initialization

We initially need to gather aerodynamic and propulsive characteristics of cruise and alternative cruise phases:

- Use the aerodynamics module to compute $C_{D0,cruise}$ and K_{cruise} for the cruise condition using $W_{0,guess}$ and the following conditions:

  `Mach = Mach_cruise`

  `altitude = altitude_cruise`

  `n_engines_failed = 0`

  `flap_def = 0.0`

  `slat_def = 0.0`

  `lg_down = 0`

  `h_ground = 0`

- Use the aerodynamics module to compute $C_{D0,altcruise}$ and $K_{altcruise}$ for the cruise condition using $W_{0,guess}$ and the following conditions:

  `Mach = Mach_altcruise`

  `altitude = altitude_altcruise`

  `n_engines_failed = 0`

  `flap_def = 0.0`

```
slat_def = 0.0
lg_down = 0
h_ground = 0
```

- Compute the engine specific fuel consumption at the cruise condition (C_{cruise}) using M_{cruise} and h_{cruise} using the TSFC module.
- Compute the engine specific fuel consumption at the alternate cruise condition ($C_{altcruise}$) using $M_{altcruise}$ and $h_{altcruise}$ using the TSFC module.

Finally, we initialize the variable M_f that will hold the product of all weight fractions.

$$M_f = 1.0 \tag{127}$$

We will overwrite its value throughout the code.

3.5.4 Engine start and warp-up

Use Fig. 6 to get the fuel fraction for this phase and then update the value of M_f. For transport jets we have:

$$M_f = M_f \cdot 0.990 \tag{128}$$

3.5.5 Taxi

Use Fig. 6 to get the fuel fraction for this phase and then update the value of M_f. For transport jets we have:

$$M_f = M_f \cdot 0.990 \tag{129}$$

3.5.6 Take-off

Use Fig. 6 to get the fuel fraction for this phase and then update the value of M_f. For transport jets we have:

$$M_f = M_f \cdot 0.995 \tag{130}$$

3.5.7 Climb

Use Fig. 6 to get the fuel fraction for this phase and then update the value of M_f. For transport jets we have:

$$M_f = M_f \cdot 0.980 \tag{131}$$

3.5.8 Cruise

Before any computation regarding the cruise phase, we need to store the current fuel fraction for posterior calculations. Therefore, define a variable named `Mf_cruise` which will store the current `Mf` value:

$$M_{f,cruise} = M_f \tag{132}$$

This will be one output of this module.

We can use the Breguet range equation to estimate the fuel consumption for the cruise phase. First we need to determine the air properties at the cruise flight level. We can use the auxiliary function provided in the `designTools.py` file for that:

```
T,p,rho,mi = atmosphere(altitude_cruise, 288.15)
```

Once we have the air temperature (T), we can determine the speed of sound (a) with:

$$a_{cruise} = \sqrt{\gamma \cdot R \cdot T} \tag{133}$$

where $\gamma = 1.4$ and $R = 287$ J/kg/K for air.

Then the aircraft cruise speed (V_{cruise}) is:

$$V_{cruise} = M_{cruise} \cdot a_{cruise} \tag{134}$$

We can also estimate the aircraft weight at the start of the cruise with:

$$W_{cruise} = M_f \cdot W_0 \tag{135}$$

Remember to use `W0_guess` as your estimate for W_0 in the equation above. We can use the $L = W_{cruise}$ condition to compute the cruise lift coefficient:

$$C_{L,cruise} = \frac{2 \cdot W_{cruise}}{\rho \cdot S_w \cdot V_{cruise}^2} \tag{136}$$

The corresponding drag coefficient can be computed with the drag polar:

$$C_{D,cruise} = C_{D0,cruise} + K_{cruise} \cdot C_{L,cruise}^2 \tag{137}$$

The cruise fuel fraction estimated by the Breguet equation is:

$$FF_{cruise} = \exp\left(-\frac{R_{cruise} \cdot C_{cruise} \cdot C_{D,cruise}}{V_{cruise} \cdot C_{L,cruise}}\right) \tag{138}$$

Now we aggregate this fuel fraction with the overall product:

$$M_f = M_f \cdot FF_{cruise} \tag{139}$$

3.5.9 Loiter

FAR 125.377 specifies that short- and medium-range aircraft should carry enough fuel to loiter for 45 minutes, while long-range aircraft should be able to loiter for 10% of the originally estimated flight time.

The aircraft will control its airspeed so that it remains at the best lift-to-drag ratio during the loiter. The maximum L/D for a parabolic drag polar is given by:

$$L/D_{\max} = \frac{1}{2 \cdot \sqrt{C_{D0,cruise} \cdot K_{cruise}}} \tag{140}$$

We assume that the drag polar coefficients will remain approximately the same between the cruise and loiter phases. However, we can consider that the engine fuel consumption will be reduced in 20% due to the different throttle settings (Raymer [3] Tab. 3.3).

$$C_{loiter} = 0.8 \cdot C_{cruise} \tag{141}$$

Now the loiter fuel fraction will be:

$$FF_{loiter} = \exp\left(-\frac{E_{loiter} \cdot C_{loiter}}{L/D_{\max}}\right) \tag{142}$$

Now we aggregate this fuel fraction with the overall product:

$$M_f = M_f \cdot FF_{loiter} \tag{143}$$

3.5.10 Descent

Use Fig. 6 to get the fuel fraction for this phase and then update the value of M_f. For transport jets we have:

$$M_f = M_f \cdot 0.990 \tag{144}$$

3.5.11 Alternate cruise

Aircraft should fly with spare fuel to let you reach an alternate airport or fly for an additional period of time (see FAR 125.377). We can use the same formulation of the cruise phase to estimate the fuel fraction of the alternate cruise.

We can use the auxiliary function provided in the `designTools.py` file to get the atmospheric properties:

```
T,p,rho,mi = atmosphere(altitude_altcruise, 288.15)
```

Once we have the air temperature (T), we can determine the speed of sound (a) with:

$$a_{altcruise} = \sqrt{\gamma \cdot R \cdot T} \tag{145}$$

where $\gamma = 1.4$ and $R = 287$ J/kg/K for air.

Then the aircraft alternate cruise speed ($V_{altcruise}$) is:

$$V_{altcruise} = M_{altcruise} \cdot a_{altcruise} \tag{146}$$

We can also estimate the aircraft weight at the start of the alternate cruise with:

$$W_{altcruise} = M_f \cdot W_0 \tag{147}$$

Remember to use `W0_guess` as your estimate for W_0 in the equation above. We can use the $L = W_{altcruise}$ condition to compute the cruise lift coefficient:

$$C_{L,altcruise} = \frac{2 \cdot W_{altcruise}}{\rho \cdot S_w \cdot V_{altcruise}^2} \tag{148}$$

The corresponding drag coefficient can be computed with the drag polar:

$$C_{D,altcruise} = C_{D0,altcruise} + K_{altcruise} \cdot C_{L,altcruise}^2 \tag{149}$$

The cruise fuel fraction estimated by the Breguet equation is:

$$FF_{altcruise} = \exp\left(-\frac{R_{altcruise} \cdot C_{altcruise} \cdot C_{D,altcruise}}{V_{altcruise} \cdot C_{L,altcruise}}\right) \tag{150}$$

Now we aggregate this fuel fraction with the overall product:

$$M_f = M_f \cdot FF_{altcruise} \tag{151}$$

3.5.12 Landing taxi and shutdown

Use Fig. 6 to get the fuel fraction for this phase and then update the value of M_f. For transport jets we have:

$$M_f = M_f \cdot 0.992 \tag{152}$$

3.5.13 Fuel weight

The variable M_f now represents the ratio between the final and the initial aircraft weights. The fuel weight is the difference between these two values. Therefore, we can use M_f to calculate the fuel weight as follows:

$$W_f = 1.06 \cdot (1 - M_f) \cdot W_0 \tag{153}$$

Once again, remember to use `W0_guess` as your estimate for W_0 in the equation above. The 6% factor refers to trapped fuel that remains in the aircraft lines (Raymer [3] Eq. 3.13).

The fuel weight (W_f) is the final output of this module.

3.5.14 Test case

Consider a test case with the following inputs:

```
Inputs of the Fuel Weight Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
'S_h': 18.196687370600415,
'S_v': 14.959999999999999,
'S_w': 93.5,
'Swet_f': 292.60345689585,
'TO_flap_def': 0.3490658503988659,
'TO_slat_def': 0.0,
'W_allelse': 79475.0,
'W_crew': 4463.55,
'W_eng': 35075.863123799056,
'W_f': 68890.55789155893,
'W_h': 4819.756583850933,
'W_mlg': 17087.125,
'W_nlg': 3015.3749999999995,
'W_payload': 95519.97,
'W_v': 3962.4552,
'W_w': 34840.475533682125,
'altitude_altcruise': 4572,
'altitude_cruise': 10668.0,
'altitude_landing': 0.0,
'altitude_takeoff': 0.0,
'b_flap_b_wing': 0.6,
'b_h': 9.18872294715571,
'b_slat_b_wing': 0.75,
'b_v': 4.358807176281144,
'b_w': 28.074988869098416,
'c_flap_c_wing': 1.2,
'c_slat_c_wing': 1.05,
'c_tank_c_w': 0.4,
'clmax_w': 2.1,
'cm_h': 2.107457619636192,
'cm_v': 3.4576757510555542,
'cm_w': 3.756317488774531,
'cr_h': 2.849393124273043,
'cr_v': 3.944978890651773,
'cr_w': 5.3933059334262,
'ct_h': 1.1112633184664868,
'ct_v': 2.919284379082312,
'ct_w': 1.267426894355157,
'dihedral_h': 0.03490658503988659,
'dihedral_w': 0.08726646259971647,
'distance_landing': 1800.0,
'distance_takeoff': 1800.0,
'eta_h': 1.0,
'flap_type': 'double slotted',
'h_ground': 10.668000000000001,
'k_exc_drag': 0.03,
'loiter_time': 2700,
'n_engines': 2,
'n_engines_under_wing': 0,
'range_altcruise': 370400.0,
'range_cruise': 2222400.0,
'rho_f': 804,
'slat_type': 'slat',
'sweep_h': 0.4537856055185257,
'sweep_v': 0.715584993317675,
'sweep_w': 0.3045599544730105,
'taper_h': 0.39,
'taper_v': 0.74,
'taper_w': 0.235,
'tcr_h': 0.1,
'tcr_v': 0.1,
'tcr_w': 0.123,
'tct_h': 0.1,
'tct_v': 0.1,
'tct_w': 0.096,
'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}

W0_guess = 467500.00000000000000
```

You should get the following results:

```
Outputs of the Fuel Weight Module

Wf = 99417.01829592324793

Mf_cruise = 0.95569551000000
```

3.6 Maximum Takeoff Weight module

This module implements the iterative process to compute the MTOW using the previous modules.

3.6.1 Inputs

The dictionary inputs of this module are:

- Weight parameters: $W_{payload}$ (W_payload), and W_{crew} (W_crew).

The function also has additional input arguments to aid in iterative procedures

- Weight parameter: W_0 (W0_guess).
- Engine parameters: T_0 (T0_guess).

3.6.2 Outputs

The outputs of this module are:

- Weight parameters: W_0 (W0), W_e (We), W_f (Wf), and $M_{f,cruise}$ (Mf_cruise).
- CG parameter: $x_{CG,e}$ (xcg_e).

This function has no specific modification to the dictionary (but it calls functions that perform modifications).

3.6.3 Maximum Takeoff Weight iteration

The MTOW should be determined iteratively due to its implicit relationships with fuel and empty weights. Algorithm 1 below describes this iterative procedure:

Algorithm 1 MTOW iteration

```
1: $\Delta = 1000$
2: while $\Delta > 10$ do
3:     Use the fuel weight module to compute $W_f$ using $W_{0,guess}$
4:     Use the empty weight module to compute $W_e$ using $W_{0,guess}$
5:     $W_0 = W_{payload} + W_{crew} + W_f + W_e$                    ▷ Recompute the MTOW
6:     $\Delta = W_0 - W_{0,guess}$      ▷ Compute the difference between the guess and computed value
7:     $W_{0,guess} = W_0$                          ▷ Set the computed value as the new guess
```

The process should converge after a few iterations. The W_0 value after the code exits the `while` statement is the converged MTOW value.

3.6.4 Test case

Consider a test case with the following inputs:

```
Inputs of the Weight Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_allelse': 79475.0,
 'W_crew': 4463.55,
 'W_eng': 35075.863123799056,
 'W_f': 68890.55789155893,
 'W_h': 4819.756583850933,
 'W_mlg': 17087.125,
 'W_nlg': 3015.3749999999995,
 'W_payload': 95519.97,
 'W_v': 3962.4552,
 'W_w': 34840.475533682125,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_v': 4.358807176281144,
 ’b_w’: 28.074988869098416,
 ’c_flap_c_wing’: 1.2,
 ’c_slat_c_wing’: 1.05,
 ’c_tank_c_w’: 0.4,
 ’clmax_w’: 2.1,
 ’cm_h’: 2.107457619636192,
 ’cm_v’: 3.4576757510555542,
 ’cm_w’: 3.756317488774531,
 ’cr_h’: 2.849393124273043,
 ’cr_v’: 3.944978890651773,
 ’cr_w’: 5.3933059334262,
 ’ct_h’: 1.1112633184664868,
 ’ct_v’: 2.919284379082312,
 ’ct_w’: 1.267426894355157,
 ’dihedral_h’: 0.03490658503988659,
 ’dihedral_w’: 0.08726646259971647,
 ’distance_landing’: 1800.0,
 ’distance_takeoff’: 1800.0,
 ’eta_h’: 1.0,
 ’flap_type’: ’double slotted’,
 ’h_ground’: 10.668000000000001,
 ’k_exc_drag’: 0.03,
 ’loiter_time’: 2700,
 ’n_engines’: 2,
 ’n_engines_under_wing’: 0,
 ’range_altcruise’: 370400.0,
 ’range_cruise’: 2222400.0,
 ’rho_f’: 804,
 ’slat_type’: ’slat’,
 ’sweep_h’: 0.4537856055185257,
 ’sweep_v’: 0.715584993317675,
 ’sweep_w’: 0.3045599544730105,
 ’taper_h’: 0.39,
 ’taper_v’: 0.74,
 ’taper_w’: 0.235,
 ’tcr_h’: 0.1,
 ’tcr_v’: 0.1,
 ’tcr_w’: 0.123,
 ’tct_h’: 0.1,
 ’tct_v’: 0.1,
 ’tct_w’: 0.096,
 ’x_mlg’: 17.8,
 ’x_n’: 23.2,
 ’x_nlg’: 3.6,
 ’x_tailstrike’: 23.68,
 ’x_tank_c_w’: 0.2,
 ’xcg_crew’: 2.5,
 ’xcg_payload’: 14.4,
 ’xm_h’: 34.21520026085125,
 ’xm_v’: 31.17587613521955,
 ’xm_w’: 15.659971822785682,
 ’xr_h’: 33.07320337042791,
 ’xr_v’: 29.25388711043971,
 ’xr_w’: 13.5,
 ’xt_h’: 35.74855563619494,
 ’xt_v’: 33.299364009371466,
 ’xt_w’: 18.944010614572072,
 ’y_mlg’: 2.47,
 ’y_n’: 2.6,
 ’ym_h’: 1.9611423076663264,
 ’ym_w’: 5.569532204800901,
 ’yt_h’: 4.594361473577855,
 ’yt_w’: 14.037494434549208,
 ’z_lg’: -2.0,
 ’z_n’: 0.0,
 ’z_tailstrike’: -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}

W0_guess = 467500.00000000000000

T0_guess = 140250.00000000000000
```

You should get the following results:

```
Outputs of the Weight Module

W0 = 428711.39823255542433

We = 237265.82251529331552

Wf = 91462.05571726211929

Mf_cruise = 0.95569551000000

xcg_e = 17.36135372255389
```

3.6.5 Homework

Do the following tasks:

1) Create a pie chart (or a similar plot) showing the weight breakdown of the test case aircraft. This chart should show the weight percentages corresponding to payload, crew, fuel, and each aircraft component.

2) Now we will take the test case and vary the wing aspect ratio (AR_w) while keeping the other parameters fixed. Generate a plot of W_0 as a function of AR_w (wing aspect ratio) for the interval $7.0 < AR_w < 14$. Make a short paragraph describing the trends of the plot (why MTOW increases or decreases as AR varies).

ATTENTION: Remember to call the *geometry* function to update the aircraft dimensions before calling the *weight* function.

3) Repeat steps 1) and 2) for the new aircraft that the group is designing.

Send the plots and discussion to ney@ita.br until the date defined in the class slides.

3.7 Performance module

This module determines the minimum thrust required to meet mission constraints.

3.7.1 Inputs

The dictionary inputs of this module are:

- Weight parameter: W_0 (`W0`).
- Wing geometric parameters: S_w (`S_w`).
- Engine parameters: n_{eng} (`n_engines`), and BPR (`BPR`).
- Flap parameters: $\delta_{flap,TO}$ (`TO_flap_def`), and $\delta_{flap,LD}$ (`LD_flap_def`).
- Slat parameters: $\delta_{slat,TO}$ (`TO_slat_def`), and $\delta_{slat,LD}$ (`LD_slat_def`).
- Ground effect parameter: h_{ground} (`h_ground`).
- Takeoff parameters: h_{TO} (`altitude_takeoff`), and d_{TO} (`distance_takeoff`).
- Landing parameters: h_{LD} (`altitude_landing`), d_{LD} (`distance_landing`), and $MLW/MTOW$ (`MLW_frac`).
- Cruise parameters: h_{cruise} (`altitude_cruise`), and M_{cruise} (`Mach_cruise`).

This function also has additional input arguments to aid in iterative procedures:

- Weight parameter: W_0 (`W0`). (after MTOW is converged)
- Cruise parameters: $M_{f,cruise}$ (`Mf_cruise`).

3.7.2 Outputs

The outputs returned by this module are:

- Engine parameters: T_0 (`T0`), and $\vec{T_0}$ (`T0vec`).
- Performance parameter: $\Delta S_{w,lan}$ (`deltaS_wlan`).
- Aerodynamics parameter: $C_{Lmax,TO}$ (`CLmaxTO`).

This function does not have specific modifications to the dictionary, but it calls functions that perform modifications.

3.7.3 Takeoff analysis

First we need to compute the atmospheric properties at the takeoff condition:

```
T,p,rho,mi = atmosphere(altitude_takeoff, 288.15)
```

Then compute the ratio σ between the current air density and the sea-level air density (which is 1.225 kg/m^3):

$$\sigma = \frac{\rho}{1.225} \tag{154}$$

Next use the aerodynamics module to compute the maximum lift coefficient at takeoff conditions ($C_{Lmax,TO}$). Use the following settings for this case:

```
Mach = 0.2
altitude = altitude_takeoff
n_engines_failed = 0
flap_def = TO_flap_def
slat_def = TO_flap_def
lg_down = 1
h_ground_takeoff = h_ground
Weight = W0
```

ATTENTION: Watch out not to overwrite the `h_ground` value we define as input for the Performance module when setting the value for the aerodynamic module.

ATTENTION: Store the $C_{Lmax,TO}$ value you get here for posterior use in the climb analysis (Sec. 3.7.6).

Then compute the thrust-to-weight ratio according to Roskam [5] (Eq. 3.8).

$$\frac{T_0}{W_0} = \frac{0.2387}{\sigma \cdot C_{Lmax,TO} \cdot d_{TO}} \cdot \frac{W_0}{S_w} \tag{155}$$

This equation is adapted to International System of Units (SI) units.

Now compute the thrust required for takeoff:

$$T_{0,TO} = \left(\frac{T_0}{W_0}\right) \cdot W_0 \tag{156}$$

3.7.4 Landing analysis

The landing process does not have a strong dependence on the available thrust. We only need to check if we have an adequate wing area for this phase.

Start by computing the atmospheric properties at the landing condition:

```
T,p,rho,mi = atmosphere(altitude_landing, 288.15)
```

Next use the aerodynamics module to compute the maximum lift coefficient at landing conditions ($C_{Lmax,LD}$). Use the following settings for this case:

```
Mach = 0.2
altitude = altitude_landing
n_engines_failed = 0
flap_def = LD_flap_def
slat_def = LD_slat_def
lg_down = 1
h_ground_landing = h_ground
Weight = W0*MLW_frac
```

ATTENTION: Store the $C_{Lmax,LD}$ value you get here for posterior use in the climb analysis (Sec. 3.7.6).

The `MLW_frac` parameter is the ratio between the Maximum Landing Weight (MLW) and MTOW. The landing weight may be limited by structural and performance constraints. We do not use the fuel fractions to estimate this ratio since the aircraft does not necessarily need to fly the entire cruise before landing. Instead, we can use historical trends from Fig. 7 to obtain values for this ratio.

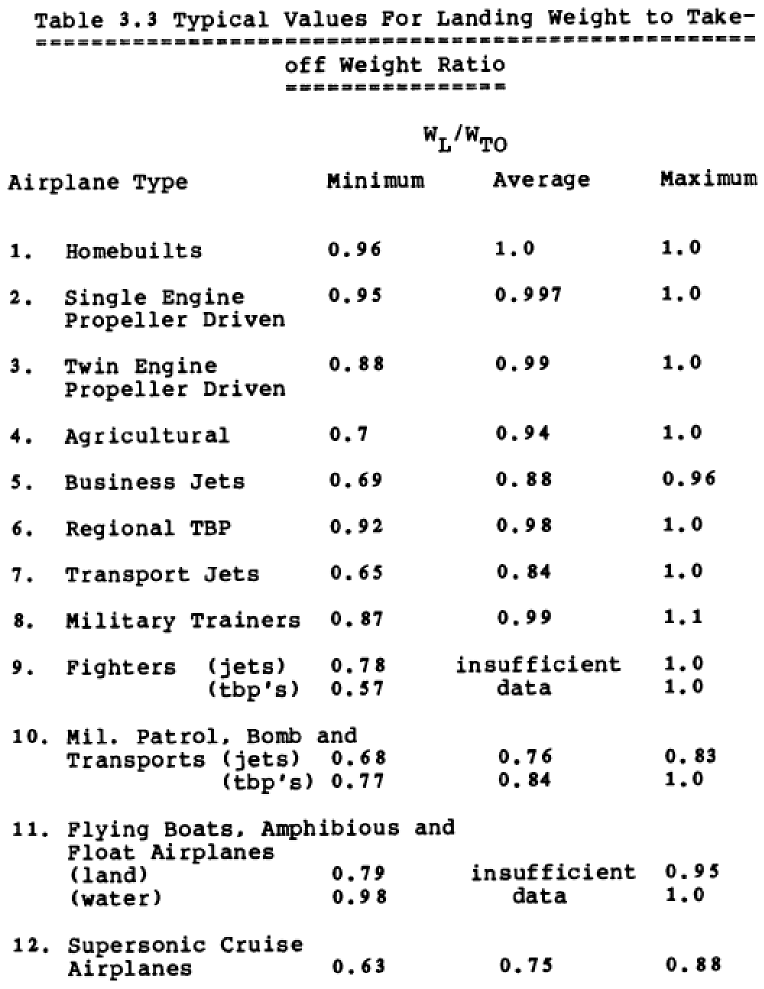


Table 3.3 Typical Values For Landing Weight to Take-off Weight Ratio

Airplane Type		W_L/W_{TO} Minimum	Average	Maximum
1. Homebuilts		0.96	1.0	1.0
2. Single Engine Propeller Driven		0.95	0.997	1.0
3. Twin Engine Propeller Driven		0.88	0.99	1.0
4. Agricultural		0.7	0.94	1.0
5. Business Jets		0.69	0.88	0.96
6. Regional TBP		0.92	0.98	1.0
7. Transport Jets		0.65	0.84	1.0
8. Military Trainers		0.87	0.99	1.1
9. Fighters	(jets)	0.78	insufficient data	1.0
	(tbp's)	0.57		1.0
10. Mil. Patrol, Bomb and Transports	(jets)	0.68	0.76	0.83
	(tbp's)	0.77	0.84	1.0
11. Flying Boats, Amphibious and Float Airplanes	(land)	0.79	insufficient data	0.95
	(water)	0.98		1.0
12. Supersonic Cruise Airplanes		0.63	0.75	0.88

Figure 7: Historical values of MLW (Roskam [5] Tab. 3.3).

The landing Balanced Field Length (BFL) can be historically correlated with the aircraft approach speed (V_a) according to the following relation (Roskam [5] Eq. 3.16):

$$V_a = 1.701 \cdot \sqrt{BFL} \tag{157}$$

This equation is adapted to SI units.

The Federal Aviation Regulations (FAR) 25 requirements establish that the approach speed should be at least 1.3 times the stall speed (V_s). Therefore the stall speed can be obtained with (Roskam [5] Eq. 3.15):

$$V_s = \frac{V_a}{1.3} \tag{158}$$

Now we can compute the required wing area to achieve this stall speed with:

$$S_{w,lan} = \frac{2 \cdot W_0 \cdot MLW/MTOW}{\rho \cdot V_s^2 \cdot C_{Lmax,LD}} \tag{159}$$

Now we only need to verify if our wing area satisfies this requirement: $S_w \geq S_{w,lan}$.
We can do this by defining a wing area excess variable $\Delta S_{w,lan}$:

$$\Delta S_{w,lan} = S_w - S_{w,lan} \tag{160}$$

A positive value of $\Delta S_{w,lan}$ indicates that the aircraft satisfies the landing requirement.

3.7.5 Cruise analysis

We need to verify if we have enough thrust to achieve the desired cruise speed. For the next derivations, we use the aircraft weight at the beginning of the cruise phase. In other words: $W_cruise = W_0 \cdot M_{f,cruise}$.

Start by computing the atmospheric properties at cruise condition:

```
T,p,rho,mi = atmosphere(altitude_cruise, 288.15)
```

Once we have the air temperature (T), we can determine the speed of sound (a) with:

$$a_{cruise} = \sqrt{\gamma \cdot R \cdot T} \tag{161}$$

where $\gamma = 1.4$ and $R = 287$ J/kg/K for air.
Then the aircraft cruise speed (V_{cruise}) is:

$$V_{cruise} = M_{cruise} \cdot a_{cruise} \tag{162}$$

We can also estimate the aircraft weight at the start of the cruise with:

$$W_{cruise} = M_f \cdot W_0 \tag{163}$$

Next use the aerodynamics module to compute the drag polar parameters at cruise conditions ($C_{D0,cruise}$ and K_{cruise}). Use the following settings for this case:

```
Mach = Mach_cruise
altitude = altitude_cruise
n_engines_failed = 0
flap_def = 0.0
slat_def = 0.0
lg_down = 0
h_ground_cruise = 0
Weight = W_cruise
```

We can use the $L = W_{cruise}$ condition to compute the cruise lift coefficient:

$$C_{L,cruise} = \frac{2 \cdot W_{cruise}}{\rho \cdot S_w \cdot V_{cruise}^2} \tag{164}$$

The corresponding drag coefficient can be computed with the drag polar:

$$C_{D,cruise} = C_{D0,cruise} + K_{cruise} \cdot C_{L,cruise}^2 \tag{165}$$

We can use the $T_{cruise} = D$ condition to find the required thrust at cruise:

$$T_{cruise} = \frac{\rho V_{cruise}^2}{2} \cdot S_w \cdot C_{D,cruise} \tag{166}$$

However, we need to revert this thrust setting back to takeoff conditions so that we have a common ground for comparison among the multiple requirements. The thrust correction factor is [6]:

$$k_T = (0.0013 \cdot BPR - 0.0397) \cdot \frac{h_{cruise}}{1000} - 0.0248 \cdot BPR + 0.7125 \tag{167}$$

The corresponding takeoff thrust demanded by the cruise condition is:

$$T_{0,cruise} = \frac{T_{cruise}}{k_T} \tag{168}$$

3.7.6 Climb analysis

Since we will have to analyze multiple climb conditions, it is recommended to implement an auxiliary function to compute the required thrust for a generic climb constraint. Define a function `climb_analysis` that receives the following inputs:

- climb gradient: γ_{climb}
- stall speed factor: k_s
- climb altitude: h_{climb} (`altitude_climb`)
- guess for maximum lift coefficient: $C_{Lmax,guess}$
- landing gear position: `lg_down`
- distance to the ground: $h_{ground,climb}$ (`h_ground_climb`)
- flap deflection: δ_{flap} (`flap_def`)
- slat deflection: δ_{slat} (`slat_def`)
- number of failed engines: $n_{eng,f}$ (`n_engines_failed`)
- weight fraction with respect to MTOW: M_f
- thrust correction factor: k_T
- MTOW: W_0
- wing area: S_w

The only output of this function is:

- required thrust at climb conditions: $T_{0,climb}$

The first set of steps will use the $C_{Lmax,guess}$ to estimate a climb Mach number. This Mach number will be used for the drag polar computation so that we can get an updated $C_{Lmax,climb}$ value.

After receiving the function inputs, determine the atmospheric properties at the climb condition:

```
T,p,rho,mi = atmosphere(altitude_climb, 288.15)
```

Once we have the air density (ρ), we can determine the aircraft stall speed with:

$$V_s = \sqrt{\frac{2 \cdot W_0 \cdot M_f}{\rho \cdot S_w \cdot C_{Lmax,guess}}} \tag{169}$$

Now we use the stall speed factor to compute the aircraft climb speed (V_{climb}):

$$V_{climb} = k_s \cdot V_s \tag{170}$$

We can use the air temperature (T) to find the speed of sound:

$$a_{climb} = \sqrt{\gamma \cdot R \cdot T} \tag{171}$$

Then the climb Mach number is:

$$M_{climb} = \frac{V_{climb}}{a_{climb}} \tag{172}$$

Next use the aerodynamics module to compute the drag polar parameters at climb conditions ($C_{D0,climb}$, K_{climb}, and $C_{Lmax,climb}$). Use the following settings for this case:

```
Mach = Mach_climb
altitude = altitude_climb
n_engines_failed = <according to the function input>
flap_def = <according to the function input>
slat_def = <according to the function input>
lg_down = <according to the function input>
h_ground = h_ground_climb
Weight = W0*Mf
```

Now that we have an updated $C_{Lmax,climb}$ value, the relationship between the climb speed and stall speed ($V_{climb} = k_s \cdot V_s$) allows us to determine the climb C_L:

$$C_{L,climb} = \frac{C_{Lmax,climb}}{k_s^2} \tag{173}$$

The corresponding drag coefficient can be computed with the drag polar:

$$C_{D,climb} = C_{D0,climb} + K_{climb} \cdot C_{L,climb}^2 \tag{174}$$

Then determine the thrust-to-weight ratio of this climb condition:

$$\left(\frac{T_0}{W_{climb}}\right) = \frac{n_{eng}}{n_{eng} - n_{eng,f}} \cdot \left(\gamma_{climb} + \frac{C_{D,climb}}{C_{L,climb}}\right) \tag{175}$$

Note that this equation will adjust the required thrust according to the number of failed engines. Now we can finally compute the required thrust with:

$$T_{0,climb} = \left(\frac{T_0}{W_{climb}}\right) \cdot \frac{W_0 \cdot M_f}{k_T} \tag{176}$$

This equation already applies the thrust correction factor (k_T). Use $k_T = 0.94$ when requirement asks for maximum continuous thrust instead of maximum takeoff thrust.

You will have to call this climb analysis function for the six cases below. Note that the climb gradient changes according to the number of engines of the aircraft (n_{eng}):

1. **FAR 25.111**
   - $\gamma_{climb} = 0.012$ (2 engines), 0.015 (3 engines), or 0.017 (4 engines)
   - $k_s = 1.2$
   - $h_{climb} = h_{takeoff}$
   - $C_{Lmax,guess} = C_{Lmax,TO}$
   - `lg_down` $= 0$ (retracted landing gear)
   - $h_{ground,climb} = h_{ground}$ (use the input of the performance module, as described in Sec. 3.7.1)
   - $\delta_{flap} = \delta_{flap,TO}$ (takeoff flaps)
   - $\delta_{slat} = \delta_{slat,TO}$ (takeoff slats)
   - $n_{eng,f} = 1$ (One Engine Inoperative (OEI) condition)
   - $M_f = 1.0$ (takeoff weight)
   - $k_T = 1.0$ (takeoff thrust)

2. **FAR 25.121a**
   - $\gamma_{climb} = 0.000$ (2 engines), 0.003 (3 engines), or 0.005 (4 engines)
   - $k_s = 1.1$
   - $h_{climb} = h_{takeoff}$
   - $C_{Lmax,guess} = C_{Lmax,TO}$
   - `lg_down` $= 1$ (extended landing gear)
   - $h_{ground,climb} = h_{ground}$ (use the input of the performance module, as described in Sec. 3.7.1)
   - $\delta_{flap} = \delta_{flap,TO}$ (takeoff flaps)
   - $\delta_{slat} = \delta_{slat,TO}$ (takeoff slats)
   - $n_{eng,f} = 1$ (OEI condition)
   - $M_f = 1.0$ (takeoff weight)
   - $k_T = 1.0$ (takeoff thrust)

3. **FAR 25.121b**
   - $\gamma_{climb} = 0.024$ (2 engines), 0.027 (3 engines), or 0.030 (4 engines)
   - $k_s = 1.2$

- $h_{climb} = h_{takeoff}$
- $C_{Lmax,guess} = C_{Lmax,TO}$
- `lg_down` $= 0$ (retracted landing gear)
- $h_{ground,climb} = 0$ (no ground effect)
- $\delta_{flap} = \delta_{flap,TO}$ (takeoff flaps)
- $\delta_{slat} = \delta_{slat,TO}$ (takeoff slats)
- $n_{eng,f} = 1$ (OEI condition)
- $M_f = 1.0$ (takeoff weight)
- $k_T = 1.0$ (takeoff thrust)

4. **FAR 25.121c**

- $\gamma_{climb} = 0.012$ (2 engines), 0.015 (3 engines), or 0.017 (4 engines)
- $k_s = 1.25$
- $h_{climb} = h_{takeoff}$
- $C_{Lmax,guess} = C_{Lmax,TO}$
- `lg_down` $= 0$ (retracted landing gear)
- $h_{ground,climb} = 0$ (no ground effect)
- $\delta_{flap} = 0.0$ (retracted flaps)
- $\delta_{slat} = 0.0$ (retracted slats)
- $n_{eng,f} = 1$ (OEI condition)
- $M_f = 1.0$ (takeoff weight)
- $k_T = 0.94$ (maximum continuous thrust)

5. **FAR 25.119**

- $\gamma_{climb} = 0.032$
- $k_s = 1.30$
- $h_{climb} = h_{landing}$
- $C_{Lmax,guess} = C_{Lmax,LD}$
- `lg_down` $= 1$ (extended landing gear)
- $h_{ground,climb} = 0$ (no ground effect)
- $\delta_{flap} = \delta_{flap,LD}$ (landing flaps)
- $\delta_{slat} = \delta_{slat,LD}$ (landing slats)
- $n_{eng,f} = 0$ (all engines operative)
- $M_f = MLW/MTOW$ (landing weight)
- $k_T = 1.0$ (takeoff thrust)

6. **FAR 25.121d**

- $\gamma_{climb} = 0.021$ (2 engines), 0.024 (3 engines), or 0.027 (4 engines)
- $k_s = 1.40$
- $h_{climb} = h_{landing}$
- $C_{Lmax,guess} = C_{Lmax,LD}$
- `lg_down` $= 1$ (extended landing gear)
- $h_{ground,climb} = 0$ (no ground effect)
- $\delta_{flap} = 0.8 * \delta_{flap,LD}$ (approach flaps)
- $\delta_{slat} = 0.8 * \delta_{slat,LD}$ (approach slats)
- $n_{eng,f} = 1$ (OEI condition)
- $M_f = MLW/MTOW$ (landing weight)
- $k_T = 1.0$ (takeoff thrust)

3.7.7 Gathering outputs

We have analyzed several performance criteria. We can gather the required thrust values in a single array, which is one of the outputs of the performance module:

$$\vec{T}_0 = \left[\; T_{0,TO} \quad T_{0,cruise} \quad T_{0,FAR\,25.111} \quad T_{0,FAR\,25.121a} \quad T_{0,FAR\,25.121b} \quad T_{0,FAR\,25.121c} \quad T_{0,FAR\,25.119} \quad T_{0,FAR\,25.121d} \;\right] \tag{177}$$

Another output of the performance module is the thrust value required by the most critical phase. We add a 5% margin for the thrust selection:

$$T_0 = 1.05 \cdot \max(\vec{T}_0) \tag{178}$$

The third output is the wing area excess due to the landing constraint ($\Delta S_{w,lan}$) computed in Sec. 3.7.4. The fourth and final output is the maximum lift coefficient at takeoff conditions ($C_{Lmax,TO}$).

3.7.8 Test case

Consider a test case with the following inputs:

```
Inputs of the Performance Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_allelse': 72882.36785331425,
 'W_crew': 4463.55,
 'W_eng': 35075.863123799056,
 'W_f': 68890.55789155893,
 'W_h': 4819.756583850933,
 'W_mlg': 15669.709088462563,
 'W_nlg': 2765.242780316923,
 'W_payload': 95519.97,
 'W_v': 3962.4552,
 'W_w': 33199.86999399066,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_v': 4.358807176281144,
 'b_w': 28.074988869098416,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'cm_h': 2.107457619636192,
 'cm_v': 3.4576757510555542,
 'cm_w': 3.756317488774531,
 'cr_h': 2.849393124273043,
 'cr_v': 3.944978890651773,
 'cr_w': 5.3933059334262,
 'ct_h': 1.1112633184664868,
 'ct_v': 2.919284379082312,
 'ct_w': 1.267426894355157,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}

W0 = 428711.39823255542433

Mf_cruise = 0.95569551000000
```

You should get the following results:

```
Outputs of the Performance Module

T0 = 120382.53870955656748

T0vec = 109848.54123604406777
        87540.01294139555830
        93838.87249094626168
        99837.34994602666120
        111624.67811898718355
        76064.91005870369554
        64110.87542348848365
        114650.03686624435068

S_wlan = 22.27741763706639

CLmaxTO = 2.37303456079851
```

3.8 Thrust matching module

We had to guess a takeoff thrust value ($T_{0,guess}$) to converge the MTOW value in Sec. 3.6. However, we had to use the converged MTOW value to determine the thrust (T_0) required to satisfy the performance requirements in Sec. 3.7, and this value will not necessarily match the guessed value. Therefore, we need an iterative process to match these thrust values, as indicated in Fig. 1. We will implement this iterative process in the current module.

3.8.1 Inputs

This function requires a dictionary with all fields defined by the previous functions.

The function also has additional input arguments to start the iterative procedure:

- Weight parameter: W_0 (`W0_guess`).
- Engine parameters: T_0 (`T0_guess`).

3.8.2 Outputs

The dictionary outputs of this module are:

- Weight parameters: W_0 (`W0`), W_e (`We`), and W_f (`Wf`).
- CG parameter: $x_{CG,e}$ (`xcg_e`).
- Engine parameters: T_0 (`T0`), and $\vec{T_0}$ (`T0vec`).
- Performance parameter: $\Delta S_{w,lan}$ (`deltaS_wlan`).
- Aerodynamics parameter: $C_{Lmax,TO}$ (`CLmaxTO`).

3.8.3 Takeoff thrust iteration

The takeoff thrust (T_0) should be determined iteratively due to its implicit relationships with the MTOW. Algorithm 2 below describes this iterative procedure:

The process should converge after a few iterations. The W_0, W_f, W_e, $x_{CG,e}$, T_0, $\vec{T_0}$, $S_{w,lan}$, and $C_{Lmax,TO}$ values after the code exits the `while` statement are the converged values that should compose the outputs of the current module.

Algorithm 2 Takeoff thrust iteration

```
1: $\Delta = 1000$                                        ▷ Set auxiliary variable to control iterative process
2: while $\Delta > 10$ do
3:     Use the MTOW module (Sec. 3.6) to compute $W_0$, $W_f$, $W_e$, and $x_{CG,e}$ using $W_{0,guess}$ and $T_{0,guess}$
4:     Use the performance module (Sec. 3.7) to compute $T_0$, $\vec{T}_0$, $\Delta S_{w,lan}$, and $C_{Lmax,TO}$ using $W_0$
5:     $\Delta = T_0 - T_{0,guess}$                        ▷ Compute the difference between the guess and computed value
6:     $T_{0,guess} = T_0$                                 ▷ Set the computed value as the new guess
7:     $W_{0,guess} = W_0$                                 ▷ Also update the MTOW guess to accelerate next iteration
```

3.8.4 Test case

Consider a test case with the following inputs:

```
Inputs of the Thrust Matching Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_allelse': 72882.36785331425,
 'W_crew': 4463.55,
 'W_eng': 35075.863123799056,
 'W_f': 68890.55789155893,
 'W_h': 4819.756583850933,
 'W_mlg': 15669.709088462563,
 'W_nlg': 2765.242780316923,
 'W_payload': 95519.97,
 'W_v': 3962.4552,
 'W_w': 33199.86999399066,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_v': 4.358807176281144,
 'b_w': 28.074988869098416,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'cm_h': 2.107457619636192,
 'cm_v': 3.4576757510555542,
 'cm_w': 3.756317488774531,
 'cr_h': 2.849393124273043,
 'cr_v': 3.944978890651773,
 'cr_w': 5.3933059334262,
 'ct_h': 1.1112633184664868,
 'ct_v': 2.919284379082312,
 'ct_w': 1.267426894355157,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}

W0_guess = 467500.00000000000000

T0_guess = 140250.00000000000000
```

You should get the following results:

```
Outputs of the Thrust Matching Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'CLmaxTO': 2.373034560798506,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'T0': 116905.51958773875,
 'T0vec': [103943.7814999834,
           85542.04301785328,
           91289.98209134683,
           96929.35159443578,
           108591.15496267364,
           74004.46014201488,
           62300.66945993366,
           111338.5900835607],
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W0': 417029.8478522959,
 'W_allelse': 70896.60884061972,
 'W_crew': 4463.55,
 'W_eng': 28712.35646954906,
 'W_f': 68890.55789155893,
 'W_h': 4819.756583850933,
 'W_mlg': 15242.77090073324,
 'W_nlg': 2689.9007471882187,
 'W_payload': 95519.97,
 'W_v': 3962.4552,
 'W_w': 32692.945285195954,
 'We': 227907.35191869608,
 'Wf': 89138.97593359985,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_v': 4.358807176281144,
 'b_w': 28.074988869098416,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'cm_h': 2.107457619636192,
 'cm_v': 3.4576757510555542,
 'cm_w': 3.756317488774531,
 'cr_h': 2.849393124273043,
 'cr_v': 3.944978890651773,
 'cr_w': 5.3933059334262,
 'ct_h': 1.1112633184664868,
 'ct_v': 2.919284379082312,
 'ct_w': 1.267426894355157,
 'deltaS_wlan': 24.218094249619313,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_e': 17.166310753037887,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}
```

3.8.5 Homework

Take the test case and vary the wing area (S_w) while keeping the other parameters fixed. Store the array of required thrust values ($\vec{T}_0$) for every wing area analyzed. Generate a plot of $\vec{T}_0$ as a function of S_w (wing area) for the interval $80.0 < S_w < 140.0$. Note that this plot should have eight curves corresponding to the eight performance requirements described in Sec. 3.7. Add a vertical line at the wing area value where $\Delta S_{w,an} = 0$ to indicate the landing constraint. Remember to add legends to identify these curves. Which performance requirements limit the thrust selection within this wing area interval?

ATTENTION: Remember to execute the `geometry` function whenever you change the wing area to resize the aircraft.

Repeat the same process for your aircraft. Remember that you may need to use a different range of wing areas. Use this plot to select a combination of wing area and thrust for your aircraft. Use the thrust value to choose an existing engine for your aircraft. Place the wing areas and thrusts of your competitors on the same plot.

Send the plots, the answers, and the code to ney@ita.br until the date defined in the class slides.

3.9 Balance module

This module will compute the position of the center of gravity and the neutral point of the aircraft. These results allow us to access the static stability of the airplane.

3.9.1 Inputs

The dictionary inputs for this module are:

- Weight parameters: W_0 (`W0`), $W_{payload}$ (`W_payload`), $x_{CG,payload}$ (`xcg_payload`), W_{crew} (`W_crew`), $x_{CG,crew}$ (`xcg_crew`), W_e (`We`), $x_{CG,e}$ (`xcg_e`), and W_f (`Wf`).
- Cruise parameter: M_{cruise} (`Mach_cruise`).
- Fuel tank parameters: c_{tank}/c_w (`c_tank_c_w`), and x_{tank}/c_w (`x_tank_c_w`).
- Wing geometric parameters: S_w (`S_w`), AR_w (`AR_w`), Λ_w (`sweep_w`), b_w (`b_w`), $x_{r,w}$ (`xr_w`), $c_{r,w}$ (`cr_w`), $c_{t,w}$ (`ct_w`), $x_{m,w}$ (`xm_w`), $c_{m,w}$ (`cm_w`), $(t/c)_{r,w}$ (`tcr_w`), and $(t/c)_{t,w}$ (`tct_w`).
- Horizontal tail geometric parameters: S_h (`S_h`), AR_h (`AR_h`), Λ_h (`sweep_h`), b_h (`b_h`), $c_{r,h}$ (`cr_h`), $c_{t,h}$ (`ct_h`), $x_{m,h}$ (`xm_h`), $c_{m,h}$ (`cm_h`), and η_h (`eta_h`).
- Vertical tail geometric parameter: C_{vt} (`Cvt`).
- Fuselage parameters: L_f (`L_f`), and D_f (`D_f`).
- Nacelle parameter: y_n (`y_n`).
- Engine parameters: T_0 (`T0`), and n_{eng} (`n_engines`).
- Aerodynamics parameter: $C_{Lmax,TO}$ (`CLmaxTO`).
- Fuel parameter: ρ_f (`rho_f`).

3.9.2 Outputs

The dictionary outputs of this module are:

- CG parameters: $x_{CG,fwd}$ (`xcg_fwd`), and $x_{CG,aft}$ (`xcg_aft`).
- Stability parameters: x_{NP} (`xnp`), SM_{fwd} (`SM_fwd`), and SM_{aft} (`SM_aft`).
- Fuel tank parameter: b_{tank}/b_w (`b_tank_b_w`).
- Control parameter: C_{Lv} (`CLv`).

3.9.3 Fuel tank center of gravity

We will consider that the fuel is stored in wing tanks. These tanks starts at the wing-fuselage intersection, and we will compute the tank span necessary to store the mission fuel. This lets us verify if the wing has enough volume to hold the required fuel.

The equations below were derived assuming that the fuel tank has an obelisk shape. The fuel tank thickness is equal to the average wing thickness.

Begin by determining the required fuel volume based on the fuel density:

$$V_f = \frac{W_f}{\rho_f \cdot g} \tag{179}$$

where g is the acceleration of gravity. Then find the average wing thickness with

$$(t/c)_w = \frac{(t/c)_{r,w} + (t/c)_{t,w}}{2} \tag{180}$$

Now determine the wing span fraction that should be occupied by the tank to store this fuel volume:

$$(b_{tank}/b_w) = \frac{3 \cdot V_f}{(c_{tank}/c_w) \cdot (t/c)_w \cdot (c_{r,w}^2 + c_{t,w}^2 + c_{r,w} \cdot c_{t,w}) \cdot b_w} \tag{181}$$

It is important to verify if $(b_{tank}/b_w) < 1.0$. Otherwise, there is not enough volume inside the wing to store all the fuel.

Next compute the lateral distance between the tank centroid and the symmetry plane:

$$y_{CG,f} = (b_{tank}/b_w) \cdot \frac{b_w}{8} \cdot \frac{c_{r,w}^2 + 2 \cdot c_{r,w} \cdot c_{t,w} + 3 \cdot c_{t,w}^2}{c_{r,w}^2 + c_{r,w} \cdot c_{t,w} + c_{t,w}^2} \tag{182}$$

Note that the tank centroid is the fuel center of gravity. The next step is the computation of the sweep angle at the tank centerline (Λ_{tank} or `sweep_tank`). We can use the auxiliary function `geo_change_sweep` for this purpose. To convert the sweep at the quarter-chord (Λ_w) to the sweep at the tank centerline we can use:

```
sweep_tank = geo_change_sweep(0.25, x_tank_c_w + c_tank_c_w/2, sweep_w, b_w/2, cr_w, ct_w)
```

Now we can finally determine the longitudinal position of the tank centerline, which is the fuel center of gravity:

$$x_{CG,f} = x_{r,w} + c_{r,w} \cdot \left((x_{tank}/c_w) + \frac{c_{tank}/c_w}{2} \right) + y_{CG,f} \cdot \tan \Lambda_{tank} \tag{183}$$

3.9.4 Center of gravity variation

In this section we analyze different loading scenarios to find the range of CG positions. These loading scenarios are:

1. Empty airplane
2. Crew
3. Payload and crew
4. Fuel and crew
5. Payload, crew, and fuel (MTOW)

The center of gravity of these scenarios are:

$$x_{CG,1} = x_{CG,e} \tag{184}$$

$$x_{CG,2} = \frac{W_e \cdot x_{CG,e} + W_{crew} \cdot x_{CG,crew}}{W_e + W_{crew}} \tag{185}$$

$$x_{CG,3} = \frac{W_e \cdot x_{CG,e} + W_{payload} \cdot x_{CG,payload} + W_{crew} \cdot x_{CG,crew}}{W_e + W_{payload} + W_{crew}} \tag{186}$$

$$x_{CG,4} = \frac{W_e \cdot x_{CG,e} + W_f \cdot x_{CG,f} + W_{crew} \cdot x_{CG,crew}}{W_e + W_f + W_{crew}} \tag{187}$$

$$x_{CG,5} = \frac{W_e \cdot x_{CG,e} + W_f \cdot x_{CG,f} + W_{payload} \cdot x_{CG,payload} + W_{crew} \cdot x_{CG,crew}}{W_0} \tag{188}$$

Now we can find the foremost and rearmost CG positions by taking the extreme values of the analyzed cases:

$$x_{CG,fwd} = \min(x_{CG,1}\ ,\ x_{CG,2}\ ,\ x_{CG,3}\ ,\ x_{CG,4}\ ,\ x_{CG,5}) \tag{189}$$

$$x_{CG,aft} = \max(x_{CG,1}\ ,\ x_{CG,2}\ ,\ x_{CG,3}\ ,\ x_{CG,4}\ ,\ x_{CG,5}) \tag{190}$$

When the aircraft if flying, we will always have the crew, payload, or fuel onboard. Therefore, the CG range of the aircraft in flight will be bounded by all scenarios except scenario 1. This lets us determine this CG range with:

$$x_{CG,fwd,flight} = \min(x_{CG,2}\ ,\ x_{CG,3}\ ,\ x_{CG,4}\ ,\ x_{CG,5}) \tag{191}$$

$$x_{CG,aft,flight} = \max(x_{CG,2}\ ,\ x_{CG,3}\ ,\ x_{CG,4}\ ,\ x_{CG,5}) \tag{192}$$

3.9.5 Neutral point

The first step is the calculation of the aerodynamic centers and lift curve slopes of the wing and tail.

We initially need to compute the wing sweep at the maximum thickness position ($\Lambda_{maxt,w}$ or `sweep_maxt_w`), which is usually located at 40% of the chord: We once again use the auxiliary function `geo_change_sweep` for this task:

```
sweep_maxt_w = geo_change_sweep(0.25, 0.40, sweep_w, b_w/2, cr_w, ct_w)
```

Next, find the square of the compressibility correction factor β:

$$\beta^2 = 1 - M_{cruise}^2 \tag{193}$$

The lift curve slope of the wing is given by (Raymer [3] Eq. 12.6):

$$C_{L\alpha,w} = \frac{2 \cdot \pi \cdot AR_w}{2 + \sqrt{4 + AR_w^2 \cdot \dfrac{\beta^2}{0.95^2} \cdot \left(1 + \dfrac{\tan^2 \Lambda_{maxt,w}}{\beta^2}\right)}} \cdot 0.98 \tag{194}$$

The wing aerodynamic center is located at the quarter-chord of the mean aerodynamic chord:

$$x_{AC,w} = x_{m,w} + \frac{c_{m,w}}{4} \tag{195}$$

The horizontal tail sweep at the maximum thickness position ($\Lambda_{maxt,h}$ or `sweep_maxt_h`) can be found with:

```
sweep_maxt_h = geo_change_sweep(0.25, 0.40, sweep_h, b_h/2, cr_h, ct_h)
```

The lift curve slope of the horizontal tail is given by (Raymer [3] Eq. 12.6):

$$C_{L\alpha,h} = \frac{2 \cdot \pi \cdot AR_h}{2 + \sqrt{4 + AR_h^2 \cdot \dfrac{\beta^2}{0.95^2} \cdot \left(1 + \dfrac{\tan^2 \Lambda_{maxt,h}}{\beta^2}\right)}} \cdot 0.98 \tag{196}$$

The horizontal tail aerodynamic center is located at the quarter-chord of the mean aerodynamic chord:

$$x_{AC,h} = x_{m,h} + \frac{c_{m,h}}{4} \tag{197}$$

The downwash factor is estimated with (Nelson [7] Eq. 2.23):

$$\frac{\partial \epsilon}{\partial \alpha} = \frac{2 \cdot C_{L\alpha,w}}{\pi \cdot AR_w} \tag{198}$$

The fuselage moment slope is given by (Raymer [3] Eq. 16.25):

$$C_{M\alpha,f} = 0.03 \cdot \frac{180}{\pi} \cdot \frac{D_f^2 \cdot L_f}{c_{m,w} \cdot S_w} \tag{199}$$

Now we can finally estimate the neutral point with (Raymer [3] Eq 16.9 and Eq 16.23):

$$x_{NP} = \frac{C_{L\alpha,w} \cdot x_{AC,w} - C_{M\alpha,f} \cdot c_{m,w} + \eta_h \cdot \frac{S_h}{S_w} \cdot C_{L\alpha,h} \cdot \left(1 - \frac{\partial \epsilon}{\partial \alpha}\right) \cdot x_{AC,h}}{C_{L\alpha,w} + \eta_h \cdot \frac{S_h}{S_w} \cdot C_{L\alpha,h} \cdot \left(1 - \frac{\partial \epsilon}{\partial \alpha}\right)} \tag{200}$$

3.9.6 Static margin

Now we can evaluate the static margin range of the aircraft by using the CG variation expected during flight:

$$SM_{fwd} = \frac{x_{NP} - x_{CG,fwd,flight}}{c_{m,w}} \tag{201}$$

$$SM_{aft} = \frac{x_{NP} - x_{CG,aft,flight}}{c_{m,w}} \tag{202}$$

3.9.7 Vertical tail lift for OEI takeoff condition

If we assume a critical case where the takeoff speed V2 is constrained both by stall speed ($V_2 > 1.2 \cdot V_{stall}$) and minimum control speed ($V_2 > 1.1V_{MC}$ from FAR 25.107), we can define the stall speed factor K_s as:

$$K_s = \frac{V_{MC}}{V_{stall}} = \frac{1.2}{1.1} \tag{203}$$

Then we can use a simplified model based on the moments of the remaining engine and the vertical tail to derive the lift coefficient that the vertical tail must generate to balance the airplane:

$$C_{Lv} = \frac{y_n}{b_w} \cdot \frac{C_{Lmax,TO}}{K_s^2} \cdot \frac{T_0}{W_0 \cdot n_{eng} \cdot C_{vt}} \tag{204}$$

Then we can check if this lift coefficient is feasible with rudder deflection.

3.9.8 Test case

Consider a test case with the following inputs:

```
Inputs of the Balance Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'CLmaxTO': 2.373034560798506,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'T0': 116905.51958773875,
 'T0vec': [103943.7814999834,
           85542.04301785328,
           91289.98209134683,
           96929.35159443578,
           108591.15496267364,
            74004.46014201488,
            62300.66945993366,
            111338.5900835607],
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W0': 417029.8478522959,
 'W_allelse': 70896.60884061972,
 'W_crew': 4463.55,
 'W_eng': 28712.35646954906,
 'W_f': 68890.55789155893,
 'W_h': 4819.756583850933,
 'W_mlg': 15242.77090073324,
 'W_nlg': 2689.9007471882187,
 'W_payload': 95519.97,
 'W_v': 3962.4552,
 'W_w': 32692.945285195954,
 'We': 227907.35191869608,
 'Wf': 89138.97593359985,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_v': 4.358807176281144,
 'b_w': 28.074988869098416,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'cm_h': 2.107457619636192,
 'cm_v': 3.4576757510555542,
 'cm_w': 3.756317488774531,
 'cr_h': 2.849393124273043,
 'cr_v': 3.944978890651773,
 'cr_w': 5.3933059334262,
 'ct_h': 1.1112633184664868,
 'ct_v': 2.919284379082312,
 'ct_w': 1.267426894355157,
 'deltaS_wlan': 24.218094249619313,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_e': 17.166310753037887,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}
```

You should get the following results:

```
Outputs of the Balance Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'CLmaxTO': 2.373034560798506,
 'CLv': 0.2941276857152721,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'SM_aft': 0.017786959718895348,
 'SM_fwd': 0.21047608790404504,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
'Swet_f': 292.60345689585,
'T0': 116905.51958773875,
'T0vec': [103943.7814999834,
          85542.04301785328,
          91289.98209134683,
          96929.35159443578,
          108591.15496267364,
          74004.46014201488,
          62300.66945993366,
          111338.5900835607],
'TO_flap_def': 0.3490658503988659,
'TO_slat_def': 0.0,
'W0': 417029.8478522959,
'W_allelse': 70896.60884061972,
'W_crew': 4463.55,
'W_eng': 28712.35646954906,
'W_f': 68890.55789155893,
'W_h': 4819.756583850933,
'W_mlg': 15242.77090073324,
'W_nlg': 2689.9007471882187,
'W_payload': 95519.97,
'W_v': 3962.4552,
'W_w': 32692.945285195954,
'We': 227907.35191869608,
'Wf': 89138.97593359985,
'altitude_altcruise': 4572,
'altitude_cruise': 10668.0,
'altitude_landing': 0.0,
'altitude_takeoff': 0.0,
'b_flap_b_wing': 0.6,
'b_h': 9.18872294715571,
'b_slat_b_wing': 0.75,
'b_tank_b_w': 0.7346738310731235,
'b_v': 4.358807176281144,
'b_w': 28.074988869098416,
'c_flap_c_wing': 1.2,
'c_slat_c_wing': 1.05,
'c_tank_c_w': 0.4,
'clmax_w': 2.1,
'cm_h': 2.107457619636192,
'cm_v': 3.4576757510555542,
'cm_w': 3.756317488774531,
'cr_h': 2.849393124273043,
'cr_v': 3.944978890651773,
'cr_w': 5.3933059334262,
'ct_h': 1.1112633184664868,
'ct_v': 2.919284379082312,
'ct_w': 1.267426894355157,
'deltaS_wlan': 24.218094249619313,
'dihedral_h': 0.03490658503988659,
'dihedral_w': 0.08726646259971647,
'distance_landing': 1800.0,
'distance_takeoff': 1800.0,
'eta_h': 1.0,
'flap_type': 'double slotted',
'h_ground': 10.668000000000001,
'k_exc_drag': 0.03,
'loiter_time': 2700,
'n_engines': 2,
'n_engines_under_wing': 0,
'range_altcruise': 370400.0,
'range_cruise': 2222400.0,
'rho_f': 804,
'slat_type': 'slat',
'sweep_h': 0.4537856055185257,
'sweep_v': 0.715584993317675,
'sweep_w': 0.3045599544730105,
'taper_h': 0.39,
'taper_v': 0.74,
'taper_w': 0.235,
'tcr_h': 0.1,
'tcr_v': 0.1,
'tcr_w': 0.123,
'tct_h': 0.1,
'tct_v': 0.1,
'tct_w': 0.096,
'x_mlg': 17.8,
'x_n': 23.2,
'x_nlg': 3.6,
'x_tailstrike': 23.68,
'x_tank_c_w': 0.2,
'xcg_aft': 17.166310753037887,
'xcg_crew': 2.5,
'xcg_e': 17.166310753037887,
'xcg_fwd': 16.160788002211415,
'xcg_payload': 14.4,
'xm_h': 34.21520026085125,
'xm_v': 31.17587613521955,
'xm_w': 15.659971822785682,
'xnp': 16.951403012174225,
'xr_h': 33.07320337042791,
'xr_v': 29.25388711043971,
'xr_w': 13.5,
'xt_h': 35.74855563619494,
'xt_v': 33.299364009371466,
'xt_w': 18.944010614572072,
'y_mlg': 2.47,
'y_n': 2.6,
'ym_h': 1.9611423076663264,
'ym_w': 5.569532204800901,
'yt_h': 4.594361473577855,
'yt_w': 14.037494434549208,
'z_lg': -2.0,
'z_n': 0.0,
'z_tailstrike': -0.84,
'zm_h': 4.42748459846653,
'zm_v': 2.070850918999471,
'zm_w': 0.4872709290626237,
'zr_h': 4.359,
'zr_v': 0.0,
'zr_w': 0.0,
'zt_h': 4.519438637980579,
'zt_v': 4.358807176281144,
'zt_w': 1.2281216273313065}
```

3.10 Landing gear module

This module analyzes landing gear position criteria.

3.10.1 Inputs

The dictionary inputs of this module are:

- Landing gear parameters: x_{nlg} (`x_nlg`), x_{mlg} (`x_mlg`), y_{mlg} (`y_mlg`), and z_{lg} (`z_lg`).
- CG parameters: $x_{CG,fwd}$ (`xcg_fwd`), and $x_{CG,aft}$ (`xcg_aft`).
- Tailstrike parameters: $x_{tailstrike}$ (`x_tailstrike`), and $z_{tailstrike}$ (`z_tailstrike`).

3.10.2 Outputs

The dictionary outputs of this module are:

- Landing gear parameters: $\eta_{nlg,fwd}$ (`frac_nlg_fwd`), $\eta_{nlg,aft}$ (`frac_nlg_aft`), $\alpha_{tipback}$ (`alpha_tipback`), $\alpha_{tailstrike}$ (`alpha_tailstrike`), and $\phi_{overturn}$ (`phi_everturn`).

3.10.3 Nose landing gear weight fraction

The weight fraction applied on the nose landing gear for the foremost CG position is:

$$\eta_{nlg,fwd} = \frac{x_{mlg} - x_{CG,fwd}}{x_{mlg} - x_{nlg}} \tag{205}$$

The weight fraction applied on the nose landing gear for the rearmost CG position is:

$$\eta_{nlg,aft} = \frac{x_{mlg} - x_{CG,aft}}{x_{mlg} - x_{nlg}} \tag{206}$$

3.10.4 Tipback angle

If we assume that the CG is along the airplane axis ($z = 0$), the tipback angle can be computed as:

$$\alpha_{tipback} = \arctan\left(\frac{x_{CG,aft} - x_{mlg}}{z_{lg}}\right) \tag{207}$$

3.10.5 Tailstrike angle

The tailstrike angle is given by:

$$\alpha_{tailstrike} = \arctan\left(\frac{z_{tailstrike} - z_{lg}}{x_{tailstrike} - x_{mlg}}\right) \tag{208}$$

3.10.6 Overturn angle

We will use the overturn angle definition according to Raymer [3] (Fig. 8).

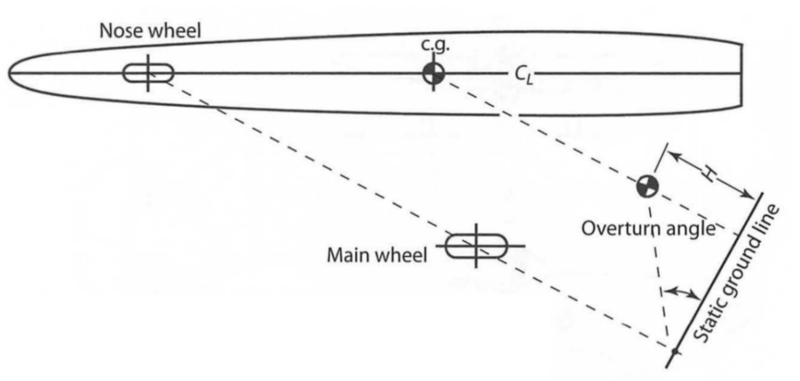


Figure 8: Overturn angle (Raymer [3] Fig. 11.5).

First we compute the static ground line distance (SGL) with:

$$SGL = \frac{(x_{CG,fwd} - x_{nlg}) \cdot y_{mlg}}{\sqrt{(x_{mlg} - x_{nlg})^2 + y_{mlg}^2}} \tag{209}$$

If we assume that the aircraft CG is on the $z = 0$ plane, then we can find the overturn angle ($\phi_{overturn}$) with:

$$\phi_{overturn} = \arctan\left(-\frac{z_{lg}}{SGL}\right) \tag{210}$$

3.10.7 Test case

Consider a test case with the following inputs:

```
Inputs of the Landing Gear Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'CLmaxTO': 2.373034560798506,
 'CLv': 0.2941276857152721,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'SM_aft': 0.017786959718895348,
 'SM_fwd': 0.21047608790404504,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'T0': 116905.51958773875,
 'T0vec': [103943.7814999834,
           85542.04301785328,
           91289.98209134683,
           96929.35159443578,
           108591.15496267364,
           74004.46014201488,
           62300.66945993366,
           111338.5900835607],
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W0': 417029.8478522959,
 'W_allelse': 70896.60884061972,
 'W_crew': 4463.55,
 'W_eng': 28712.35646954906,
 'W_f': 68890.55789155893,
 'W_h': 4819.756583850933,
 'W_mlg': 15242.77090073324,
 'W_nlg': 2689.9007471882187,
 'W_payload': 95519.97,
 'W_v': 3962.4552,
 'W_w': 32692.945285195954,
 'We': 227907.35191869608,
 'Wf': 89138.97593359985,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_tank_b_w': 0.7346738310731235,
 'b_v': 4.358807176281144,
 'b_w': 28.074988869098416,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'cm_h': 2.107457619636192,
 'cm_v': 3.4576757510555542,
 'cm_w': 3.756317488774531,
 'cr_h': 2.849393124273043,
 'cr_v': 3.944978890651773,
 'cr_w': 5.3933059334262,
 'ct_h': 1.1112633184664868,
 'ct_v': 2.919284379082312,
 'ct_w': 1.267426894355157,
 'deltaS_wlan': 24.218094249619313,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_aft': 17.166310753037887,
 'xcg_crew': 2.5,
 'xcg_e': 17.166310753037887,
 'xcg_fwd': 16.160788002211415,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xnp': 16.951403012174225,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}
```

You should get the following results:

```
Outputs of the Landing Gear Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'CLmaxTO': 2.373034560798506,
 'CLv': 0.2941276857152721,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'SM_aft': 0.017786959718895348,
 'SM_fwd': 0.21047608790404504,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'T0': 116905.51958773875,
 'T0vec': [103943.7814999834,
           85542.04301785328,
           91289.98209134683,
           96929.35159443578,
           108591.15496267364,
           74004.46014201488,
           62300.66945993366,
           111338.5900835607],
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W0': 417029.8478522959,
 'W_allelse': 70896.60884061972,
 'W_crew': 4463.55,
 'W_eng': 28712.35646954906,
 'W_f': 68890.55789155893,
 'W_h': 4819.756583850933,
 'W_mlg': 15242.77090073324,
 'W_nlg': 2689.9007471882187,
 'W_payload': 95519.97,
 'W_v': 3962.4552,
 'W_w': 32692.945285195954,
 'We': 227907.35191869608,
 'Wf': 89138.97593359985,
 'alpha_tailstrike': 0.1947777647825633,
 'alpha_tipback': 0.3068380491961717,
 'altitude_altcruise': 4572,
 ’altitude_cruise’: 10668.0,
 ’altitude_landing’: 0.0,
 ’altitude_takeoff’: 0.0,
 ’b_flap_b_wing’: 0.6,
 ’b_h’: 9.18872294715571,
 ’b_slat_b_wing’: 0.75,
 ’b_tank_b_w’: 0.7346738310731235,
 ’b_v’: 4.358807176281144,
 ’b_w’: 28.074988869098416,
 ’c_flap_c_wing’: 1.2,
 ’c_slat_c_wing’: 1.05,
 ’c_tank_c_w’: 0.4,
 ’clmax_w’: 2.1,
 ’cm_h’: 2.107457619636192,
 ’cm_v’: 3.4576757510555542,
 ’cm_w’: 3.756317488774531,
 ’cr_h’: 2.849393124273043,
 ’cr_v’: 3.944978890651773,
 ’cr_w’: 5.3933059334262,
 ’ct_h’: 1.1112633184664868,
 ’ct_v’: 2.919284379082312,
 ’ct_w’: 1.267426894355157,
 ’deltaS_wlan’: 24.218094249619313,
 ’dihedral_h’: 0.03490658503988659,
 ’dihedral_w’: 0.08726646259971647,
 ’distance_landing’: 1800.0,
 ’distance_takeoff’: 1800.0,
 ’eta_h’: 1.0,
 ’flap_type’: ’double slotted’,
 ’frac_nlg_aft’: 0.04462600330719108,
 ’frac_nlg_fwd’: 0.11543746463299898,
 ’h_ground’: 10.668000000000001,
 ’k_exc_drag’: 0.03,
 ’loiter_time’: 2700,
 ’n_engines’: 2,
 ’n_engines_under_wing’: 0,
 ’phi_overturn’: 0.7486786878113462,
 ’range_altcruise’: 370400.0,
 ’range_cruise’: 2222400.0,
 ’rho_f’: 804,
 ’slat_type’: ’slat’,
 ’sweep_h’: 0.4537856055185257,
 ’sweep_v’: 0.715584993317675,
 ’sweep_w’: 0.3045599544730105,
 ’taper_h’: 0.39,
 ’taper_v’: 0.74,
 ’taper_w’: 0.235,
 ’tcr_h’: 0.1,
 ’tcr_v’: 0.1,
 ’tcr_w’: 0.123,
 ’tct_h’: 0.1,
 ’tct_v’: 0.1,
 ’tct_w’: 0.096,
 ’x_mlg’: 17.8,
 ’x_n’: 23.2,
 ’x_nlg’: 3.6,
 ’x_tailstrike’: 23.68,
 ’x_tank_c_w’: 0.2,
 ’xcg_aft’: 17.166310753037887,
 ’xcg_crew’: 2.5,
 ’xcg_e’: 17.166310753037887,
 ’xcg_fwd’: 16.160788002211415,
 ’xcg_payload’: 14.4,
 ’xm_h’: 34.21520026085125,
 ’xm_v’: 31.17587613521955,
 ’xm_w’: 15.659971822785682,
 'xnp': 16.951403012174225,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}
```

4 Full analysis

Now that we have all modules, we can perform a complete aircraft analysis cycle.

4.1 Inputs

The dictionary inputs for the entire framework are:

- Wing geometric parameters: S_w (`S_w`), AR_w (`AR_w`), λ_w (`taper_w`), Λ_w (`sweep_w`), δ_w (`dihedral_w`), $x_{r,w}$ (`xr_w`), $z_{r,w}$ (`zr_w`), $(t/c)_{r,w}$ (`tcr_w`), and $(t/c)_{t,w}$ (`tct_w`).
- Wing airfoil parameters: $c_{lmax,w}$ (`clmax_w`).
- Horizontal tail geometric parameters: C_{ht} (`Cht`), $L_h/c_{m,w}$ (`Lc_h`), AR_h (`AR_h`), λ_h (`taper_h`), Λ_h (`sweep_h`), δ_h (`dihedral_h`), $z_{r,h}$ (`zr_h`), $(t/c)_{r,h}$ (`tcr_h`), $(t/c)_{t,h}$ (`tct_h`), and η_h (`eta_h`).
- Vertical tail geometric parameters: C_{vt} (`Cvt`), L_v/b_w (`Lb_v`), AR_v (`AR_v`), λ_v (`taper_v`), Λ_v (`sweep_v`), and $z_{r,v}$ (`zr_v`), $(t/c)_{r,v}$ (`tcr_v`), and $(t/c)_{t,v}$ (`tct_v`).
- Fuselage parameters: L_f (`L_f`), and D_f (`D_f`).
- Nacelle parameters: x_n (`x_n`), y_n (`y_n`), z_n (`z_n`), L_n (`L_n`), and D_n (`D_n`).
- Engine parameters: n_{eng} (`n_engines`), $n_{eng,w}$ (`n_engines_under_wing`), and BPR (`BPR`).
- Landing gear parameters: x_{nlg} (`x_nlg`), x_{mlg} (`x_mlg`), y_{mlg} (`y_mlg`), and z_{lg} (`z_lg`).
- Tailstrike parameters: $x_{tailstrike}$ (`x_tailstrike`), and $z_{tailstrike}$ (`z_tailstrike`).
- Fuel tank parameters: c_{tank}/c_w (`c_tank_c_w`), and x_{tank}/c_w (`x_tank_c_w`).
- Flap parameters: $\delta_{flap,TO}$ (`TO_flap_def`), $\delta_{flap,LD}$ (`LD_flap_def`), `flap_type`, c_{flap}/c_w (`c_flap_c_wing`), and b_{flap}/b_w (`b_flap_b_wing`).
- Slat parameters: $\delta_{slat,TO}$ (`TO_slat_def`), $\delta_{slat,LD}$ (`LD_slat_def`), `slat_type`, c_{slat}/c_w (`c_slat_c_wing`), and b_{slat}/b_w (`b_slat_b_wing`).
- Ground effect parameter: h_{ground} (`h_ground`).
- Excrescence drag parameter: k_{exc} (`k_exc_drag`).

- Guessed parameters: $W_{0,guess}$ (W0_guess) and $T_{0,guess}$ (T0_guess).
- Takeoff parameters: h_{TO} (altitude_takeoff), and d_{TO} (distance_takeoff).
- Landing parameters: h_{LD} (altitude_landing), d_{LD} (distance_landing), and $MLW/MTOW$ (MLW_frac).
- Cruise parameters: h_{cruise} (altitude_cruise), M_{cruise} (Mach_cruise), and R_{cruise} (range_cruise).
- Loiter parameter: E_{loiter} (loiter_time).
- Alternative cruise parameters: $h_{altcruise}$ (altitude_altcruise), $M_{altcruise}$ (Mach_altcruise), and $R_{altcruise}$ (range_altcruise).
- Weight parameters: $W_{payload}$ (W_payload), $x_{CG,payload}$ (xcg_payload), W_{crew} (W_crew), and $x_{CG,crew}$ (xcg_crew).
- Fuel parameter: ρ_f (rho_f).

4.2 Outputs

The dictionary outputs of this module are:

- Weight parameters: W_0 (W0) and W_f (Wf).
- Engine parameter: T_0 (T0).
- Performance parameter: $\Delta S_{w,lan}$ (deltaS_wlan).
- Stability parameters: x_{NP} (xnp), SM_{fwd} (SM_fwd), and SM_{aft} (SM_aft).
- Fuel tank parameter: b_{tank}/b_w (b_tank_b_w).
- Control parameter: C_{Lv} (CLv).
- Landing gear parameters: $\eta_{nlg,fwd}$ (frac_nlg_fwd), $\eta_{nlg,aft}$ (frac_nlg_aft), $\alpha_{tipback}$ (alpha_tipback), $\alpha_{tailstrike}$ (alpha_tailstrike), and $\phi_{overturn}$ (phi_everturn).

4.3 Joining all modules

After receiving all inputs, you can call the modules in the following order to go through the entire analysis cycle:

1. Geometry module
2. Thrust matching module
3. Balance module
4. Landing gear module

HINT: The airplane dictionary is modified even if it is an input of the function. Therefore, you only need to place one function call after another. There is no need to create multiple dictionaries.

This completes the entire analysis cycle. Here are some constraints that may be important when analyzing outputs:

- deltaS_wlan ≥ 0
- SM_fwd ≤ 0.30
- SM_aft ≥ 0.05
- CLv ≤ 0.75
- frac_nlg_fwd ≤ 0.18
- frac_nlg_aft ≥ 0.05
- alpha_tipback ≥ 15 deg
- alpha_tailstrike ≥ 10 deg
- phi_overturn ≤ 63 deg

4.3.1 Test case

Consider a test case with the following inputs:

```
Inputs of the Full Analysis Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'S_w': 93.5,
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W_crew': 4463.55,
 'W_payload': 95519.97,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_slat_b_wing': 0.75,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_crew': 2.5,
 'xcg_payload': 14.4,
 'xr_w': 13.5,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0}
```

You should get the following results:

```
Outputs of the Full Analysis Module

airplane = {'AR_h': 4.64,
 'AR_v': 1.27,
 'AR_w': 8.43,
 'BPR': 3.04,
 'CLmaxTO': 2.373034560798506,
 'CLv': 0.2941276857152721,
 'Cbase': None,
 'Cht': 0.94,
 'Cvt': 0.088,
 'D_f': 3.3,
 'D_n': 1.5,
 'LD_flap_def': 0.6981317007977318,
 'LD_slat_def': 0.0,
 'L_f': 32.5,
 'L_n': 4.3,
 'Lb_v': 0.55,
 'Lc_h': 4.83,
 'MLW_frac': 0.9228915662650602,
 'Mach_altcruise': 0.4,
 'Mach_cruise': 0.73,
 'SM_aft': 0.017786959718895348,
 'SM_fwd': 0.21047608790404504,
 'S_h': 18.196687370600415,
 'S_v': 14.959999999999999,
 'S_w': 93.5,
 'Swet_f': 292.60345689585,
 'T0': 116905.51958773875,
 'T0vec': [103943.7814999834,
           85542.04301785328,
           91289.98209134683,
           96929.35159443578,
           108591.15496267364,
           74004.46014201488,
           62300.66945993366,
           111338.5900835607],
 'TO_flap_def': 0.3490658503988659,
 'TO_slat_def': 0.0,
 'W0': 417029.8478522959,
 'W_allelse': 70896.60884061972,
 'W_crew': 4463.55,
 'W_eng': 28712.35646954906,
 'W_f': 68890.55789155893,
 'W_h': 4819.756583850933,
 'W_mlg': 15242.77090073324,
 'W_nlg': 2689.9007471882187,
 'W_payload': 95519.97,
 'W_v': 3962.4552,
 'W_w': 32692.945285195954,
 'We': 227907.35191869608,
 'Wf': 89138.97593359985,
 'alpha_tailstrike': 0.1947777647825633,
 'alpha_tipback': 0.3068380491961717,
 'altitude_altcruise': 4572,
 'altitude_cruise': 10668.0,
 'altitude_landing': 0.0,
 'altitude_takeoff': 0.0,
 'b_flap_b_wing': 0.6,
 'b_h': 9.18872294715571,
 'b_slat_b_wing': 0.75,
 'b_tank_b_w': 0.7346738310731235,
 'b_v': 4.358807176281144,
 'b_w': 28.074988869098416,
 'c_flap_c_wing': 1.2,
 'c_slat_c_wing': 1.05,
 'c_tank_c_w': 0.4,
 'clmax_w': 2.1,
 'cm_h': 2.107457619636192,
 'cm_v': 3.4576757510555542,
 'cm_w': 3.756317488774531,
 'cr_h': 2.849393124273043,
 'cr_v': 3.944978890651773,
 'cr_w': 5.3933059334262,
 'ct_h': 1.1112633184664868,
 'ct_v': 2.919284379082312,
 'ct_w': 1.267426894355157,
 'deltaS_wlan': 24.218094249619313,
 'dihedral_h': 0.03490658503988659,
 'dihedral_w': 0.08726646259971647,
 'distance_landing': 1800.0,
 'distance_takeoff': 1800.0,
 'eta_h': 1.0,
 'flap_type': 'double slotted',
 'frac_nlg_aft': 0.04462600330719108,
 'frac_nlg_fwd': 0.11543746463299898,
 'h_ground': 10.668000000000001,
 'k_exc_drag': 0.03,
 'loiter_time': 2700,
 'n_engines': 2,
 'n_engines_under_wing': 0,
 'phi_overturn': 0.7486786878113462,
 'range_altcruise': 370400.0,
 'range_cruise': 2222400.0,
 'rho_f': 804,
 'slat_type': 'slat',
 'sweep_h': 0.4537856055185257,
 'sweep_v': 0.715584993317675,
 'sweep_w': 0.3045599544730105,
 'taper_h': 0.39,
 'taper_v': 0.74,
 'taper_w': 0.235,
 'tcr_h': 0.1,
 'tcr_v': 0.1,
 'tcr_w': 0.123,
 'tct_h': 0.1,
 'tct_v': 0.1,
 'tct_w': 0.096,
 'x_mlg': 17.8,
 'x_n': 23.2,
 'x_nlg': 3.6,
 'x_tailstrike': 23.68,
 'x_tank_c_w': 0.2,
 'xcg_aft': 17.166310753037887,
 'xcg_crew': 2.5,
 'xcg_e': 17.166310753037887,
 'xcg_fwd': 16.160788002211415,
 'xcg_payload': 14.4,
 'xm_h': 34.21520026085125,
 'xm_v': 31.17587613521955,
 'xm_w': 15.659971822785682,
 'xnp': 16.951403012174225,
 'xr_h': 33.07320337042791,
 'xr_v': 29.25388711043971,
 'xr_w': 13.5,
 'xt_h': 35.74855563619494,
 'xt_v': 33.299364009371466,
 'xt_w': 18.944010614572072,
 'y_mlg': 2.47,
 'y_n': 2.6,
 'ym_h': 1.9611423076663264,
 'ym_w': 5.569532204800901,
 'yt_h': 4.594361473577855,
 'yt_w': 14.037494434549208,
 'z_lg': -2.0,
 'z_n': 0.0,
 'z_tailstrike': -0.84,
 'zm_h': 4.42748459846653,
 'zm_v': 2.070850918999471,
 'zm_w': 0.4872709290626237,
 'zr_h': 4.359,
 'zr_v': 0.0,
 'zr_w': 0.0,
 'zt_h': 4.519438637980579,
 'zt_v': 4.358807176281144,
 'zt_w': 1.2281216273313065}
```

4.4 Homework

Have you noticed that the test case aircraft may have a very small static margin? One way of solving this is by redistributing internal components to change the CG position. We can simulate this by changing the CG position of the "all-else" component (Eq. (124)) within the bounds of Fig. 5.

Which fraction you should use for the "all-else" CG to satisfy the constrained mentioned at the end of Sec. 4.3.1? Send your answer and the outputs of the analysis cycle to ney@ita.br until the beginning of the next class.

5 Report instructions

Now you can use the analysis module developed throughout the course to design an airplane according to the requirements given in the Course Outline. There are other design choices that are not included in this sequence of modules, but this framework will help you perform some trade-studies.

ATTENTION: The values used in the test cases are just reference values. They should NOT be used as starting point for your new design. It may be wise to check historical trends for your initial analyses.

Each group has to showcase their design in a report with at most 40 pages.

DUE DATE: Sept 29th, 2021. 11:59 PM.

The design decisions and the quality of the report will be considered for the grading. The report should cover, at least, the following topics:

- Introduction
  - Design requirements
  - Aircraft overview
- Design process
  - Configuration selection
  - Trade-off studies
  - Design refinement (iterations) if applicable

  - Comparison against competitors

- Geometry and layout
  - Cabin cross section
  - Cabin floorplan
  - Three-views with major dimensions
- Aerodynamics
  - Planform selection (non-dimensional parameters)
  - Drag polars for takeoff, landing, and cruise configurations
  - High-lift devices
- Weights
  - Weight distribution
  - Fuel tanks
  - CG location
- Performance
  - T-S diagram
  - Critical mission phases
  - Requirements vs Actual performance
- Propulsion
  - Engine selection
  - Engine characteristics
  - Engine location
- Structures
  - Wing structural layout
  - Tail structural layout
  - Fuselage structural layout
- Stability and Controls
  - Tail design
  - Static stability
  - Control surfaces
- Landing gear
  - Landing gear location
  - Landing gear in retracted position
  - Weight distribution and critical angles
- Conclusion

You are free to change the report road map. You do not need to follow this list as is.
You can get some bonus points if you bring studies that were not mentioned in the previous list. Here are some examples:

- Airfoil selection
- Spanwise lift distribution
- V-n Diagram
- Payload-Range diagram
- List of systems

6 OpenVSP

The Master student Filipi Kunz recorded a nice series of videos explaining how to use OpenVSP to generate the aircraft geometry. This may be helpful to obtain the three-views of the aircraft (although the use of this software is not required for this class).

1. Download e visão geral do software - `https://youtu.be/PEJio4HZc7w`
2. Configuração da Asa - `https://youtu.be/Ux4oeTDjn64`
3. Configuração da fuselagem - `https://youtu.be/Y5AygQPvcoQ`
4. Modelagem de objetos conformáveis (+ linkagens) - `https://youtu.be/cfdv2O9cP4U`
5. Modelagem de componentes internos - `https://youtu.be/a5VOhbQjZaQ`
6. Modelagem de componentes externos - `https://youtu.be/FfEZHfyWr-M`
7. Avaliação das propriedades de massa - `https://youtu.be/DrGYhJfvH6Y`
8. Modelagem do Boeing 737-800 - `https://youtu.be/79SbTgJu0_I`
9. Passo a passo da configuração do OpenVSP Python API - `https://youtu.be/jpirajddG3o`

Most videos refer to RC aircraft, but the 8^{th} video may be helpful for commercial aircraft.

Constants

The following constants where used in the test cases:

- acceleration of gravity: `g = 9.81`
- conversion from feet to meters: `ft2m = 0.3048`
- conversion from knots to meters per second: `kt2ms = 0.514444`
- conversion from pounds to Newtons: `lb2N = 4.44822`

List of code variables

All coordinates (x, y, and z) use the fuselage node as origin of the system.

`S_w`	wing area
`AR_w`	wing aspect ratio
`taper_w`	wing taper ratio
`sweep_w`	wing sweep at quarter-chord
`dihedral_w`	wing dihedral
`xr_w`	longitudinal position of the wing root leading edge
`zr_w`	vertical position of the wing root leading edge
`Cht`	horizontal tail volume coefficient
`AR_h`	horizontal tail aspect ratio
`taper_h`	horizontal tail taper ratio
`sweep_h`	horizontal tail sweep at quarter-chord
`dihedral_h`	horizontal tail dihedral
`Lc_h`	horizontal tail lever ratio
`zr_h`	vertical position of the horizontal tail root leading edge
`Cvt`	vertical tail volume coefficient
`AR_v`	vertical tail aspect ratio
`taper_v`	vertical tail taper ratio
`sweep_v`	vertical tail sweep at quarter-chord
`Lb_v`	vertical tail lever ratio
`zr_v`	vertical position of the vertical tail root leading edge
`b_w`	wing span
`cr_w`	wing root chord

`xt_w`	longitudinal position of the wing tip leading edge
`yt_w`	lateral position of the wing tip leading edge
`zt_w`	vertical position of the wing tip leading edge
`ct_w`	wing tip chord
`xm_w`	longitudinal position of the wing mean aerodynamic chord leading edge
`ym_w`	lateral position of the wing mean aerodynamic chord leading edge
`zm_w`	vertical position of the wing mean aerodynamic chord leading edge
`cm_w`	wing mean aerodynamic chord
`clmax_w`	2D maximum lift coefficient of the wing airfoil
`S_h`	horizontal tail area
`b_h`	horizontal tail span
`xr_h`	longitudinal position of the horizontal tail root leading edge
`cr_h`	horizontal tail root chord
`xt_h`	longitudinal position of the horizontal tail tip leading edge
`yt_h`	lateral position of the horizontal tail tip leading edge
`zt_h`	vertical position of the horizontal tail tip leading edge
`ct_h`	horizontal tail tip chord
`xm_h`	longitudinal position of the horizontal tail mean aerodynamic chord leading edge
`ym_h`	lateral position of the horizontal tail mean aerodynamic chord leading edge
`zm_h`	vertical position of the horizontal tail mean aerodynamic chord leading edge
`cm_h`	horizontal tail mean aerodynamic chord
`S_v`	vertical tail area
`b_v`	vertical tail span
`xr_v`	longitudinal position of the vertical tail root leading edge
`cr_v`	vertical tail root chord
`xt_v`	longitudinal position of the vertical tail tip leading edge
`zt_v`	vertical position of the vertical tail tip leading edge
`ct_v`	vertical tail tip chord
`xm_v`	longitudinal position of the vertical tail mean aerodynamic chord leading edge
`zm_v`	vertical position of the vertical tail mean aerodynamic chord leading edge
`cm_v`	vertical tail mean aerodynamic chord
`tcr_w`	thickness ratio of the wing root airfoil
`tct_w`	thickness ratio of the wing tip airfoil
`tcr_h`	thickness ratio of the horizontal tail root airfoil
`tct_h`	thickness ratio of the horizontal tail tip airfoil
`tcr_v`	thickness ratio of the vertical tail root airfoil
`tct_v`	thickness ratio of the vertical tail tip airfoil
`L_f`	fuselage length
`D_f`	fuselage diameter
`L_n`	nacelle length
`D_n`	nacelle diameter
`x_n`	longitudinal position of the center of the nacelle forward face
`y_n`	lateral position of the center of the nacelle forward face
`z_n`	vertical position of the center of the nacelle forward face
`Mach`	flight Mach number
`altitude`	flight altitude
`n_engines`	total number of engines of the aircraft
`n_engines_under_wing`	number of engines installed under the wing
`n_engines_failed`	number of inoperative engines
`flap_def`	flap deflection
`TO_flap_def`	takeoff flap deflection
`LD_flap_def`	landing flap deflection
`flap_type`	flap type
`c_flap_c_wing`	total chord with extended flaps divided by total chord with retracted flaps
`b_flap_b_wing`	fraction of the wing span with flaps
`slat_def`	slat deflection
`TO_slat_def`	takeoff slat deflection
`LD_slat_def`	landing slat deflection
`slat_type`	slat type
`c_slat_c_wing`	total chord with extended slats divided by total chord with retracted slats

b_slat_b_wing	fraction of the wing span with slats
lg_down	flag to indicate if landing gear is lowered
h_ground	distance to the ground for ground effect computation
k_exc_drag	excrescence drag factor
W0_guess	guess for the MTOW
W0	converged MTOW
CD0	parasite drag coefficient
K	induced drag factor
CLmax	maximum lift coefficient
Swet_f	fuselage wetted area
CD0_cruise	parasite drag coefficient at cruise
K_cruise	induced drag factor at cruise
altitude_cruise	cruise altitude
Mach_cruise	cruise Mach number
range_cruise	cruise range
C_cruise	engine specific fuel consumption at cruise [1/s]
loiter_time	loiter time [s]
CD0_altcruise	parasite drag coefficient at alternative cruise
K_altcruise	induced drag factor at alternative cruise
altitude_altcruise	alternative cruise altitude
Mach_altcruise	alternative cruise Mach number
range_altcruise	alternative cruise range
C_altcruise	engine specific fuel consumption at alternative cruise [1/s]
Wf	fuel weight
Mf_cruise	product of weight factors up to the beginning of the cruise
Mf	ratio between aircraft weight at a given phase and the MTOW
T0_guess	guess for the total takeoff thrust (considering all engines)
BPR	engine bypass ratio
x_nlg	longitudinal position of the nose landing gear
x_mlg	longitudinal position of the main landing gear
y_mlg	lateral position of the main landing gear
z_lg	vertical position of the extended landing gear
We	empty weight
xcg_e	longitudinal position of the empty aircraft center of gravity
C	specific fuel consumption
W_payload	payload weight
xcg_payload	longitudinal position of the payload center of gravity
W_crew	crew weight
xcg_crew	longitudinal position of the crew center of gravity
altitude_takeoff	takeoff altitude
distance_takeoff	takeoff field length
altitude_landing	landing altitude
distance_landing	landing field length
MLW_frac	ratio between MLW and MTOW
T0	takeoff thrust required to satisfy all mission phases
T0vec	takeoff thrust required by each mission phase
deltaS_wlan	excess of wing area for the landing requirement
eta_h	dynamic pressure loss at the tails
c_tank_c_w	fraction of the wing chord occupied by the fuel tank
x_tank_c_w	fraction of the wing chord where fuel tank starts
rho_f	fuel density
xcg_fwd	foremost longitudinal position of the aircraft center of gravity
xcg_aft	rearmost longitudinal position of the aircraft center of gravity
xnp	longitudinal position of the aircraft neutral point
SM_fwd	static margin for the foremost CG position
SM_aft	static margin for the rearmost CG position
b_tank_b_w	fraction of the wing span occupied by the fuel tank
CLv	vertical tail lift coefficient required for OEI takeoff
x_tailstrike	longitudinal position of the critical tailstrike point
z_tailstrike	vertical position of the critical tailstrike point

`frac_nlg_fwd`	weight fraction on the nose landing gear for foremost CG position
`frac_nlg_aft`	weight fraction on the nose landing gear for rearmost CG position
`alpha_tipback`	tipback angle
`alpha_tailstrike`	tailstrike angle
`phi_overturn`	overturn angle

List of Acronyms

BFL Balanced Field Length

FAR Federal Aviation Regulations

MLW Maximum Landing Weight

MTOW Maximum Takeoff Weight

OEI One Engine Inoperative

SI International System of Units

References

[1] Torenbeek, E., *Synthesis of subsonic aircraft design*, Springer Netherlands, 1982.

[2] Howe, D., *Aircraft conceptual design synthesis*, Professional Engineering Publishing London, UK, 2000.

[3] Raymer, D. P., *Aircraft Design: A Conceptual Approach*, 5th ed., American Institute of Aeronautics and Astronautics, Inc., 2012.

[4] Roskam, J., *Aircraft Design. Part II: Preliminary Configuration Design and Integration of the Propulsion System*, Roskam Aviation and Engineering Corporation, Kansas, 1985.

[5] Roskam, J., *Aircraft Design. Part I: Preliminary Sizing of Airplanes*, Roskam Aviation and Engineering Corporation, Kansas, 1985.

[6] Scholz, D., "Preliminary Sizing," , 2020. URL https://www.fzt.haw-hamburg.de/pers/Scholz/HOOU/AircraftDesign_5_PreliminarySizing.pdf, accessed: 2020-07-27.

[7] Nelson, R. C., *Flight Stability and Automatic Control*, 2nd ed., WCB/McGraw Hill New York, 1998.